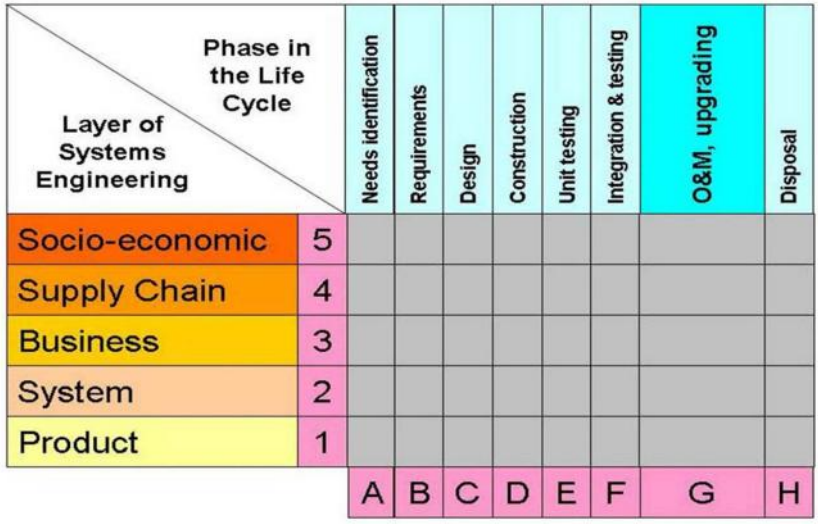
A Framework for Understanding Systems Engineering



REVISED AND UPDATED

DR JOSEPH KASSER

NUS Student version [F34]

A Framework for Understanding Systems Engineering

Dr Joseph E. Kasser
CEng, CM, FIET, CMALT, G3ZCZ

Second edition 2013
First edition 2007

First edition produced by Dr Joseph Kasser with the assistance of a grant from the Leverhulme Trust to Cranfield University.

Published by The Right Requirement
50 Crane Way, Cranfield, Bedfordshire, MK43 0HH, England.

ISBN-13: 978-1482758160
ISBN-10: 1482758164

Visit https://www.createspace.com/1482758164 to order additional copies.

Dedication

To Annie, Belle, Brock, Daliah, Jesse, Martha, Molly and Wendy, who were there for me when I needed them.

Acknowledgments

Co-authors of the original papers upon which some of the Chapters are based or quoted at length – Dave Denzler, Stephen Cook, Tim Ferris, Moti Frank, Derek K. Hitchins, Thomas V. Huynh, Angus Massie, Sharon Shoshany, Kent Palmer, Xuan-Linh Tran, Victoria R. Williams and Yang Yang Zhao.

Friends who reviewed the manuscript and made constructive comments, Piet Beukman, Amihud Hari, and Mary J Simpson.

Previously by Joseph E. Kasser
(on the subject of systems engineering)

Holistic Thinking: Creating innovative solutions to complex problems, Createspace, 2013.

Applying Total Quality Management to Systems Engineering, Artech House, 1996.

About the author

Joseph Kasser has been a practicing systems engineer for 40+ years and an academic for about 14 years. He is a Fellow of the Institution of Engineering and Technology (IET), an INCOSE Fellow, the author of "A Framework for Understanding Systems Engineering" and "Applying Total Quality Management to Systems Engineering" and many conference and symposia papers. He is a recipient of the National Aeronautical and Space Administration's (NASA) Manned Space Flight Awareness Award (Silver Snoopy) for quality and technical excellence for performing and directing systems engineering and other awards. He holds a Doctor of Science in Engineering Management from The George Washington University. He is a Certified Manager and holds a Certified Membership of the Association for Learning Technology. He has performed and directed systems engineering in the UK, United States of America (US), Israel and Australia. He gave up his positions as a Deputy Director and DSTO Associate Research Professor at the Systems Engineering and Evaluation Centre at the University of South Australia in early 2007 to move to the UK to develop the world's first immersion course in systems engineering as a Leverhulme Visiting Professor at Cranfield University. He is currently a Visiting Associate Professor at the National University of Singapore.

Preface to Second Edition

The research into the nature of systems engineering discussed in the collected papers in the first edition did not stop with the publication of the volume. It split into two areas:

- **Systems thinking** which is a critical skill needed by systems engineers. This research picked up Kline's dictum quoted in Section 18.4 and the statement that systems engineering can be considered as being documented in the literature on critical thinking, systems thinking, problem solving in the activities that take place in the Area of Concern discussed in Section 21.4.1 and determined the need to go beyond systems thinking to create innovative solutions to complex problems. The research produced the nine holistic thinking perspectives (HTP) publishing along the way (Kasser and Mackley, 2008; Kasser, 2009; Kasser, 2011a). The research also investigated problem-solving (Kasser, 2010; 2011b; Kasser, 2012), one of the major activities performed by systems engineers. These papers are not included in this volume because they have been expanded into a practical guide on how to create innovative solutions to complex problems and published as a separate volume (Kasser, 2013), see https://www.createspace.com/4171581.
- **Systems engineering** which continued studying systems engineering from different perspectives including developing a framework for benchmarking competencies of systems engineers described in Chapter 24. Chapter 7 was inserted into this edition as a precursor to Chapter 24 and because it is cited several times in the book. This research is documented in papers now updated, and included in Chapters 22 to 29. The research concluded that systems engineering is an enabling discipline which provides a set of thinking tools used in many different activities in the same way that mathematics is an enabling discipline providing a set of calculating tools used in many different activities as described in Chapter 29.

In preparing this manuscript, many of the typographical and formatting errors that crept into the first edition have been corrected.

The research has not stopped with the publication of this updated edition. It has focused on developing better ways of teaching systems engineering, and providing teaching case studies to help students grasp the principles of systems engineering more effectively.

Contents

Figures

Tables

2013

1 Introduction

Systems engineering started as an approach to solving the complex problems facing the defence and aerospace industries in the middle of the 20th century (Johnson, 1997). However it has since evolved to become a preferred approach for identifying and solving the problems faced in the course of acquiring and maintaining the complex systems that underpin modern 21st century civilization. Consequently, the demand for systems engineers is growing and it is becoming increasingly recognised as providing a career path to the top of the organization.

However, if you pose the question "what is systems engineering?" to a number of systems engineers, you will get a different answer from each of them. The reason for this situation is that systems engineering is still not a recognised engineering discipline but is evolving into one as documented in this book. Even so, training as a systems engineer will allow you to develop skills and competencies for successful careers in many different industries. Why is this? Well while systems engineering has been associated with high technology and computers, its concepts are just as applicable in other settings. For example, as a musical metaphor, playing an instrument is akin to computer programming, composing a symphony is akin to software engineering, but orchestrating the performance and ensuring that all the necessary resources for the performance to take place and conducting the symphony is akin to systems engineering. In short, in becoming a systems engineer:

- You will develop an understanding of the technology underpinning our modern civilization. You will also develop an understanding of the relationship between the components both technical and non-technical that make up the systems upon which our lifestyles depend.
- You will develop skills that will allow you to focus on the real problem in any situation and apply appropriate tools and methodologies for providing a solution to the problem, especially non-textbook problems.

This book[1]:

[1] The use of the dot point format is deliberate. It enables you to quickly identify and read pertinent information.

- Will not make you a systems engineer. It will however, help you understand the nature of systems engineering.
- Applies the holistic thinking (Kasser, 2013) to the process, the product being produced and the organisational environment in which the process is producing the product.
- Views systems engineering from various perspectives including management and quality, and offers suggestions for improvement based on the authors' experience and the literature.
- Offers complimentary and sometimes conflicting conclusions from the different views of systems engineering leading to the hypothesis that the problem of understanding systems engineering may indeed be a wicked problem (Rittel and Webber, 1973) discussed in Chapter 21.
- Documents progress along an unfinished journey of research and exploration into the nature and application of systems engineering that began more than 15 years ago[2]. The core content of each Chapter in this book has already been published in the proceedings of International conferences on systems engineering, software engineering and engineering management, and has since been updated. Where the topic was covered by more than one paper, the papers have been combined to form the Chapter. The original papers and the Chapters in this book are correlated in Table 1-1.
- Brings the material together for the first time for a wider audience than the conference participants, namely systems and software engineers, managers, educators of systems and software engineers and managers, and even aspiring systems and software engineers and managers is designed to make you think about systems engineering and project management, develop an understanding of their nature and be able to tailor them to the specific situation you find yourself in to optimize the process, the product being produced and the organizational environment.

The development, operations and maintenance of systems and software are performed in an organisation. Represent that organisation by the functions drawn within a boundary as shown in Figure 1-1. The functions are grouped as project management, systems engineering, test and evaluation (T&E) sometimes also known as Quality, and a host of other functions (e.g. process improvement, marketing, sales, support, etc.). Note that as in many organisations, there is no clear separation between the functions in the fig-

[2] As you progress through the book, you will find references to earlier Chapters (papers at the time of publication) illustrating how the research progressed. Footnotes may comment on the content in the light of later research and provide a forward reference to a subsequent Chapter.

ure; the reason for this depiction will become clear as you read further into the book.

Table 1-1 Cross Reference from Chapter to Conference Paper

Chapter	Based on
2	Kasser J. E., “Systems Engineering: Myth or Reality”, proceedings of the 6th Annual Symposium of the International Council on Systems Engineering (INCOSE), Boston, MA, 1996.
3	Kasser, J. E., "There's No Place for Managers in a Quality Organization", proceedings of the Ninth Annual National Conference on Federal Quality, 1996.
	Kasser, J. E., "Applying Systems Engineering to the Organization: Announcing the Excellence Paradigm", proceedings of the Portland International Conference on Management of Engineering and Technology (PICMET), 1997.
4	Kasser J. E., “Transition via Transactions: First steps in creating a customer driven organization”, proceedings of the First World Customer Service Congress, Tyson's Corner, VA, 1997.
5	Kasser J. E., “What Do You Mean, You Can't Tell Me How Much of My Project Has Been Completed?” proceedings of the 7th Annual International Symposium of the INCOSE, Los Angeles, CA, 1997.
	Kasser J. E., “Yes Virginia, You Can Build a Defect Free System, On Schedule and Within Budget”, proceedings of the 7th Annual International Symposium of the INCOSE, Los Angeles, CA, 1997.
	Kasser, J. E., “Using Organizational Engineering to Build Defect Free Systems, On Schedule and Within Budget”, proceedings of the PICMET, Portland OR, 1999.
6	Kasser J. E., Williams V.R., “What Do You Mean You Can’t Tell Me If My Project Is in Trouble?” proceedings of the First European Conference on Software Metrics (FESMA 98), Antwerp, Belgium, 1998.
7	Kasser J.E., “The Certified Systems Engineer - It's About Time!” proceedings of the Systems Engineering, Test and Evaluation Conference 2000 (SETE 2000), Brisbane, Australia, 2000.

8 Kasser J.E., “A Framework for Requirements Engineering in a Digital Integrated Environment (FREDIE)”, proceedings of the Systems SETE 2000 Conference, Brisbane, Australia, 2000.

9 Kasser J. E., “Enhancing the Role of Test and Evaluation in the Acquisition Process to Increase the Probability of the Delivery of Equipment that Meets the Needs of the Users”, proceedings of the SETE 2000 Conference, Brisbane, Australia, 2000.

10 Kasser J. E., Shoshany S., “Systems Engineers are from Mars, Software Engineers are from Venus”, proceedings of the 13th International Conference on Software and Systems Engineering and their Applications, Paris, France, 2000.

Kasser J. E., Shoshany S., “Bridging the Communications Gap between Systems and Software Engineering”, proceedings of the INCOSE-UK Spring Symposium, 2001.

11 Kasser J. E., “Writing Requirements for Flexible Systems”, proceedings of the INCOSE-UK Spring Symposium, 2001.

12 Kasser J. E., Massie A., “A Framework for a Systems Engineering Body of Knowledge”, proceedings of the 11th Annual International Symposium of the INCOSE, Melbourne, Australia, 2001.

13 Kasser J. E., “The Cataract Methodology for Systems and Software Acquisition”, proceedings of the SETE 2002 Conference, Sydney, Australia, 2002.

Kasser J. E., “Configuration Management: The silver bullet for cost and schedule control”, proceedings of the International Engineering Management Conference, Cambridge, UK, 2002.

Denzler D., Kasser J. E., “Designing Budget Tolerant Systems”, proceedings of the 5th Annual International Symposium of the National Council on Systems Engineering (NCOSE), St Louis, MO, 1995.

14 Kasser J. E, “The acquisition of a System of Systems is just a simple multi-phased parallel-processing paradigm”, proceedings of the International Engineering Management Conference (IEMC) 2002, Cambridge, UK, 2002.

Kasser J. E., “Isn’t the acquisition of a System of Systems just a Simple Multi-phased Parallel Processing Paradigm”, pro-

ceedings of the INCOSE-UK Spring Symposium, Maldon, Essex, 2002.

15 Kasser J. E., “Systems Engineering: An Alternative Management Paradigm”, proceedings of the SETE 2002 Conference, Sydney, Australia, 2002.

16 Kasser J. E., “Does Object-Oriented System Engineering Eliminate the Need for Requirements?” proceedings of the 12th Annual International Symposium of the INCOSE, Las Vegas, NV, 2002.

17 Kasser J. E., “Object-Oriented Requirements Engineering and Management”, proceedings of the SETE 2003 Conference, Canberra, Australia, 2003.

18 Kasser J. E., Palmer K., “Reducing and Managing Complexity by Changing the Boundaries of the System”, proceedings of the Conference on Systems Engineering Research (CSER), Hoboken NJ, 2005.

Kasser, J. E., “Using Organizational Engineering to Build Defect Free Systems, On Schedule and Within Budget”, proceedings of the PICMET, Portland OR, 1999.

19 Kasser J. E., “Introducing the Role of Process Architecting”, proceedings of the 15th Annual International Symposium of the INCOSE, Rochester NY. 2005.

20 Kasser, J. E., “Eight deadly defects in systems engineering and how to fix them”, proceedings of the 17th Annual International Symposium of the INCOSE, San Diego, CA, 2007.

21 Kasser, J. E., “A Proposed Framework for a Systems Engineering Discipline”, proceedings of the CSER, Hoboken, NJ, 2007.

Kasser, J. E., “The Hitchins-Kasser-Massie Framework for Systems Engineering”, proceedings of the 17th Annual International Symposium of the INCOSE, San Diego, CA., 2007.

22 Kasser, J. E., "Luz: From Light to Darkness: Lessons learned from the solar system", proceedings of the 18th Annual International Symposium of the INCOSE, Utrecht, Holland, 2008.

23 Kasser, J. E., Hitchins, D. and Huynh, T. V., “Reengineering Systems Engineering“, proceedings of the 3rd Annual Asia-Pacific Conference on systems Engineering (APCOSE), Singapore, 2009.

24 Kasser, J. E., Hitchins, D. K., Frank, M. and Zhao, Y. Y., "A framework for benchmarking competency assessment models", Systems Engineering: The Journal of the INCOSE Volume 16, No. 1, 2013.

25 Kasser, J. E., and Hitchins, D. K., "Unifying the different systems engineering processes", proceedings of CSER, Hoboken, NJ., 2010.

26 Kasser, J. E., "Seven systems engineering myths and the corresponding realities", proceedings of the SETE 2010 conference, Adelaide, Australia, 2010.

27 Kasser, J. E., and Hitchins, D. K., "Unifying systems engineering: Seven principles for systems engineered solution systems", proceedings of the 21st Annual International Symposium of the INCOSE, Denver, 2011.

28 Kasser, J. E., "Getting the right requirements right", proceedings of the 22nd Annual International Symposium of the INCOSE, Rome, Italy, 2012.

29 Kasser, J. E., and Hitchins, D. K., "Yes systems engineering, you are a discipline", proceedings of the 22nd Annual International Symposium of the INCOSE, Rome, Italy, 2012.

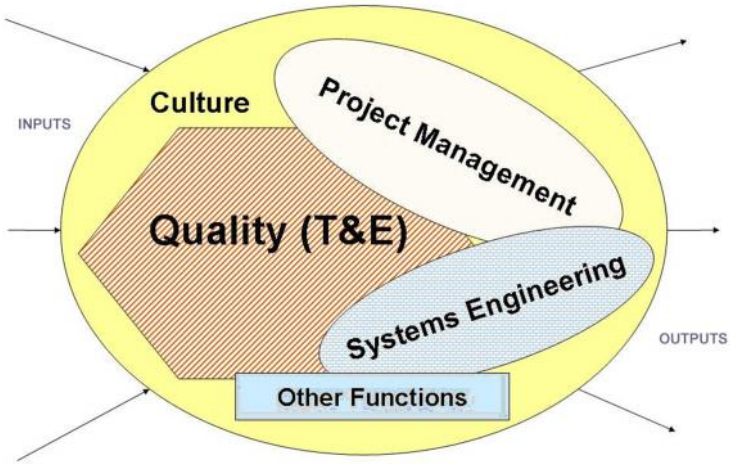


Figure 1-1 Overlapping functions in an organisation

The development, operations and maintenance of systems and software are performed in an organisation. Represent that organisation by the func-

tions drawn within a boundary as shown in Figure 1-1. The functions are grouped as project management, systems engineering, T&E sometimes also known as Quality, and a host of other functions (e.g. process improvement, marketing, sales, support, etc.). Note that as in many organisations, there is no clear separation between the functions in the figure; the reason for this depiction will become clear as you read further into the book.

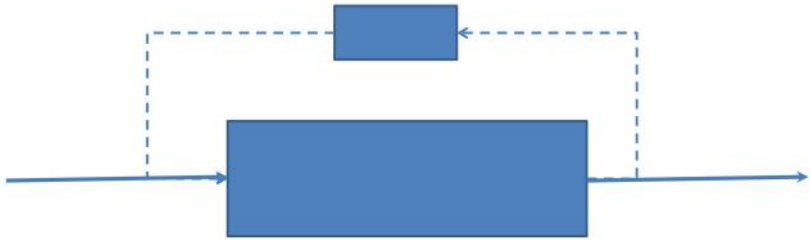


Figure 1-2 Traditional black box representation

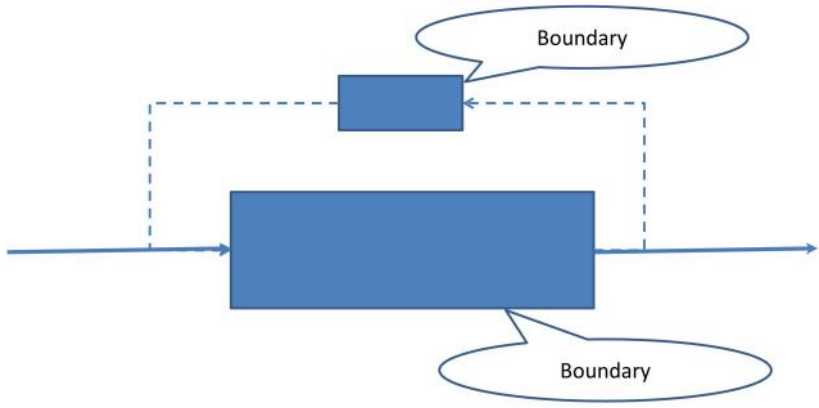


Figure 1-3 Systems are defined by their boundaries

Now the system represented in Figure 1-1 can also be represented as the traditional black box shown in Figure 1-2. What is the difference between the two representations? The answer is they show different perspectives. Figure 1-1 represents some of the internal functions of the organisation[3] while Figure 1-2 explicitly shows the system as a black box with a feedback control element. So how many systems are represented in Figure 1-2 ? The answer is "it depends". What does it depend on? It depends on where

[3] Figure 1-3Figure 1-1is a type of representation known as a "white box" since it shows some of the internal functionality of the box, as opposed to a "black box" which hides (abstracts out) the contents of the box as in Figure 1-2.

the boundaries are drawn. In Figure 1-1 and Figure 1-2 the boundary is implicitly drawn to include both boxes. However, Figure 1-3 points out that a boundary exists around each of the boxes. Thus Figure 1-3 can represent two separate coupled systems or two sub-systems of a single system. These figures demonstrate that the internal and external boundaries of a system are determined by the viewpoint of the observer and different perspectives can be obtained from the different viewpoints.

Each of the remaining Chapters in this book examines systems engineering from a different perspective as follows.

- Chapter 2 looks at the organisation from the perspective of the systems engineering function.
- Chapter 3 rotates the perspective and views the organisation from the viewpoint of process improvement.
- Chapter 4 also views the organisation from the perspective of process improvement.
- Chapter 5 looks at the organisation from the perspective of the Software and SDLC for large systems which can take several years to complete.
- Chapter 6 looks at the context or background to the Systems Development Lifecycle (SDLC). Anecdotal evidence suggests that most projects do not fail due to the non-mitigation of technical risks. Rather, they fail as a result of poor management of the human element.
- Chapter 7 looks at systems engineering from the perspective of the people who perform it, namely the systems engineers.
- Chapter 8 views the SDLC as a production system.
- Chapter 9 views the organisation from the perspective of T&E.
- Chapter 10 views the organisation from the perspective of its culture – the people who comprise the organisation and how they communicate.
- Chapter 11 views the SDLC from the perspective of the product being produced.
- Chapter 12 views systems engineering from the perspective of developing a body of knowledge for teaching the subject.
- Chapter 13 views the organisation from the process perspective.
- Chapter 14 provides an alternative perspective to much of the research effort being expended in an effort to develop new concepts that can be used to solve the problem of managing Systems of Systems.
- Chapter 15 picks up from Chapter 2 and takes another look at the organisation from the perspective of the systems engineering function.
- Chapter 16 uses the product viewpoint to examine system engineering and object-oriented methodologies.
- Chapter 17 takes a process-product object-oriented perspective on requirements engineering.

- Chapter 18 views the system from the perspective that systems are defined by their boundaries.
- Chapter 19 examines the system from the perspective of the work done in the development of systems.
- Chapter 20 points out eight of the defects in the current systems engineering paradigm that have been identified in the research.
- Chapter 21 rounds off this phase of the research by introducing a framework for understanding systems engineering.
- Chapter 22 looks at the relationships between requirements and functions in the context of a case study.
- Chapter 23 looks at the activities performed in the early stages of the SDLC and the types of systems engineers that should be in leadership positions in those stages.
- Chapter 24 picks up from Chapter 11 and the types of systems engineers identified in Chapter 23 and looks at the competencies of systems engineers.
- Chapter 25 looks at the processes that are associated with systems engineering.
- Chapter 26 discusses seven myths of systems engineering and shows the nature of the myth and the reality, and explains how and why each myth arose.
- Chapter 27 bypasses the problem of trying to gain consensus on the nature of systems engineering by providing seven principles for systems engineered solution systems.
- Chapter 28 takes another look at requirements and identifies that there are two different requirements paradigms in systems engineering.
- Chapter 29 completes the story that began in Chapter 2 and shows that systems engineering is a discipline, it is an enabling discipline used in all domains and disciplines.
- Section 30 contains a postscript to the book.
- Section 31 contains a list of acronyms used in this book.
- Section 32 contains the collected references cited in the various papers. Note the works citied are from a number of domains including psychology, software, project management, innovations, Quality, systems thinking as well as systems engineering.
- Section 33 contains the index to help you locate words used in the book.

1996

2 Systems engineering: myth or reality

This Chapter looks at the organisation from the perspective of the systems engineering function. It first unsuccessfully attempts to define systems engineering then examines the overlap between systems engineering and project management and determines that a new organisational paradigm is needed. The research to produce this Chapter was the first phase of a journey which began formally in 1994 when I noticed at the National Council on Systems Engineering (NCOSE) Symposium that lots of dedicated people spent a lot of energy assessing, measuring and educating people about an incomplete body of knowledge (systems engineering). The incompleteness was due to the lack of a definition of what that body of knowledge is supposed to cover. Now every systems engineer knows that it is important to capture all the requirements as early as possible in the program, so why had the systems engineers not defined (captured the requirements for) systems engineering? This situation led me to hypothesize that there was no such thing as systems engineering (after all, if the experts in NCOSE couldn't come up with one, then there wasn't one).

This Chapter analyses the functions performed by systems engineers, shows there seems to be no unique body of knowledge to systems engineering, which provides one reason why the NCOSE have failed to define systems engineering. The Chapter then looks outside the systems engineering box for the reasons for the failure, and provides a surprising conclusion.

2.1 Defining systems engineering

Hill and Warfield wrote "*development of a theory of systems engineering that will be broadly accepted is much to be desired*" (Hill and Warfield, 1972). Twenty years later systems engineers still had a problem, not only explaining what they do to other people, but also defining it amongst themselves. For example, at the 1994 and 1995 annual Symposia of the NCOSE, presenter after presenter opened their presentation with a definition of systems engi-

neering and each definition was different[4]. Moreover, when each presenter continued by describing the functions performed by systems engineers, they talked about planning, organizing, directing and measuring; which have long been recognised as the traditional functions of management. When asked what systems engineers did, their answers were also different, and now some 12 years further on nothing has changed, there is still no theory of systems engineering, not is there a definition that everyone can agree with. According to some, systems engineering was developed in the United States of America (US) communications industry to meet the networking challenges of the 1950's (Hall, 1962) and might even have begun in Germany in the 1940's (Mackey and Mackey, 1994). So for more than 50 years, no systems engineer has come up with a definition of systems engineering that those systems engineers can agree upon. Thus the question is

"What is there about systems engineering that defies definition"?

In an attempt determine the nature of the problem, the proceedings published at the NCOSE Symposia in 1994 and 1995 were scanned to determine if their subject matter could provide an answer. All authors seemed to agree that systems engineering was performed throughout the SDLC which spans the range of product inception, through design, development, operations and obsolescence. The number of papers on methodology and process showed there was no "standard" process for systems engineering. While papers in other engineering conferences tended to focus on applications (outward), the NCOSE 1994 and 1995 papers focused inward on:

- systems engineering methodology and process (inward);
- the early phases of product inception, namely "requirements engineering".

The insight obtained from this research was that systems engineering seemed to be in the state electrical engineering was in before the adoption of "Ohm's law"[5]. An attempt to identify such an "Ohm's law" in terms of cost elements was then initiated. The approach was to consider the activities in the SDLC from the perspective of planning and doing. The activities that pertained to planning were considered as "delays" or "lags" and the activities involved in working as "leading". These are the electrical analogues to capacitance (lags) and inductance (leads). Assuming that working without a plan is wasteful, and that if too much time is spent planning, there will be insufficient resources for actually doing the work, for any phase in the

[4] See Table 12-1 for a selection of other definitions in the literature.
[5] Or the state of chemistry before the periodic table of elements was discovered to show how the various elements related to each other.

SDLC, for a specific size of project, the optimal cost to perform the phase is the "right mix" of planning and doing as shown in Figure 2-1. This attempt failed since all the activities identified overlapped those of "project management". This led to a review of project management.

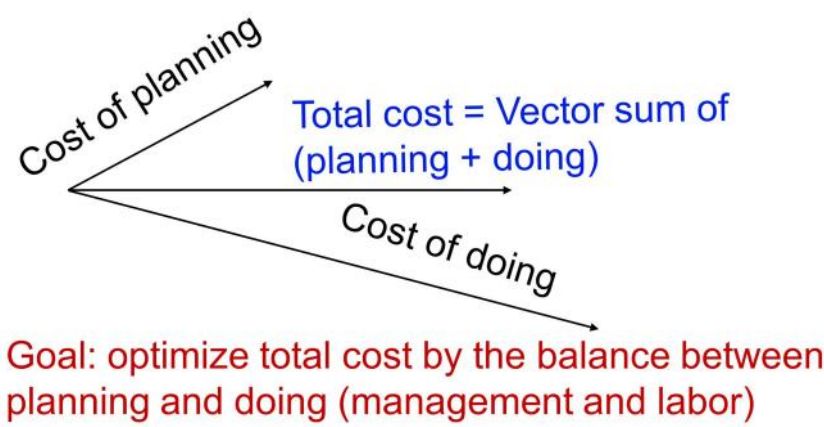


Figure 2-1 Optimal Cost of SDLC Phase

2.2 Management and systems engineering

According to Kezsbom et al. project management is defined as "*the planning, organizing, directing, and controlling of company resources (i.e. money, materials, time and people) for a relatively short-term objective. It is established to accomplish a set of specific goals and objectives by utilizing a fluid, systems approach to management by having functional personnel (the traditional line-staff hierarchy) assigned to a specific project (the horizontal hierarchy)*" (Kezsbom, et al., 1989). Kezsbom's systematic approach to project management requires the breakdown and identification of each logical subsystems component into its own assemblage of people, things, information or organization required to achieve the sub-objective. However, the role of project management seems to overlap the role of systems engineering since the role of the systems engineer is to ensure the delivery of the best working system that can be built within the constraints of schedule, budget and resources. According to Roe the knowledge and skills of systems engineering are the same as those of project management in the areas of management expertise, technical breadth and technical depth (Roe, 1995). Rose' difference is in the application where the system engineer has more technical breadth, while the project manager has more management expertise. As such, systems engineers:

- Have the responsibility for the global technical design, optimal implementation and proper verification of the system over the entire SDLC (Kasser, 1995).
- Manage subcontractors, requirements, prepare plans for systems engineering and risk management (Brekka, et al., 1994).

For systems engineers to be the only functional skill which fulfils this role, they must have a unique body of knowledge. Looking around, reviewing experience and the literature, apart from "requirements and interface engineering" there is no unique body of knowledge for systems engineering. Even the 'ilities' are careers in themselves, and have their own literature.

Arthur D. Hall however provided the following specifications or traits for an "Ideal Systems Engineer" in which the body of knowledge is only a small part. His specifications are grouped in several areas as (Hall, 1962) pages 16-18):

- An ability to see the big picture.
- Objectivity.
- Creativity.
- Human Relations.
- A Broker of Information.
- Education - graduate training in the relevant field of interest (application), as well as courses in probability and statistics, philosophy, economics, psychology, and language.
- Experience in research, development, systems engineering and operations.

Hall concluded by stating that the ideal is not available because the scope of the task is beyond the capabilities of a single individual, mixed teams of specialists and generalists are used.

2.3 Outside the box

Since looking inside systems engineering isn't providing a definition of systems engineering, look outside systems engineering. Consider the environment in which systems engineering is performed, namely the organization. The organization is configured in a hierarchical structure. The division of work between manager and worker within the western corporate organizational structure is based on adaptive modifications to "Scientific Management" (Taylor, 1911), and we have further added "Quality" as another area of endeavour as shown in Figure 2-2. The optimal management method is said to be "Management by Walking Around" (MBWA) (Peters and Austin, 1985). Yet Deming wrote *"MBWA is hardly ever effective. The reason is that someone in management, walking around, has little idea about what questions to ask, and usually does not pause long enough at any spot to get the*

right answer" (Deming, 1986) page 22).

Juran was quoted by Harrington stated that 80 to 85% of all problems are caused by management (Harrington, 1995) page 198). We spend a lot of organizational energy mitigating the effect of poor management instead of on productive work. For example, we create jobs which compensate for the lack of skills in management. One such job is the 'facilitator' who keeps meetings on track; a second example may be the 'systems engineer'. This working around ineffective managers leads to excessive complexity within the organization[6]. So within our organisations, these days, Taylor's:

Figure 2-2 Three streams of work

- **Assumptions are no longer valid** - Taylor's paradigm was for an organization in which the workers did not want to work (Theory X) (McGregor, 1960). Today most system and software development organizations are Theory Y and the workers want to get the job done.
- **Rules for the division of labour are not being followed** - Taylor split the work into a partnership between brain and brawn. Taylor assumed managers knew more about the work than did the worker. According to Taylor, managers are supposed to plan and direct activities, while the workers do the work. Yet today much of management in today's systems and software development organisations has little idea of the technical aspects of the work and consequently little idea of the technical impact of their decisions. The Dilbert cartoons, typify middle management in these types of organisations (Adams, 2006).

  Other symptoms that the Taylor organizational paradigm is broken are:

[6] Excessive complexity is a symptom of an underlying problem within the foundation of the current paradigm (Chapter 3).

- The ineffective use of promotions since financial rewards is equated with greater degrees of control. Since the skills required to be a technical person are different to those of a manager, we tend to promote a good technical person into a poor manager. Once the horizontal line in Figure 2-3 is crossed, there is no retreat within the organization, and good technical people may be lost to the organization. This particular path also leads to the impression that managers are more important than the technical personnel who produce the products which bring in the revenue.

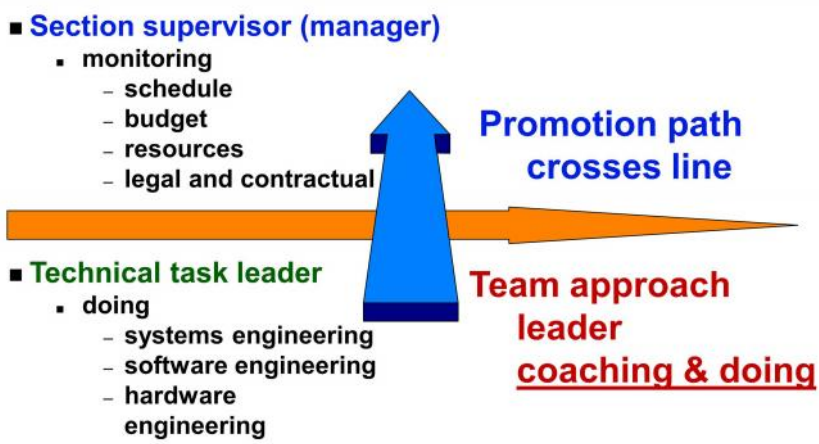


Figure 2-3 The line of no-return

- The adoption of project management and other sub-organizations which cross the boundaries of management, development and quality (Integrated Process Teams (IPTs) and concurrent engineering).
- Current symbology uses boxes for a hierarchical organization structure and circles for a process as shown in Figure 2-4. Truly a case of attempting to insert a square peg into a round hole!
- The report producing and information filtering functions of middle management have largely been replaced by technology (Rodgers, et al., 1993).

2.4 Concurrent engineering, Total Quality Management, et al.

Looking at industry today, Hall's mixed teams of systems engineers (Hall, 1962) seem to be called IPTs and are working in the context of "concurrent engineering" which has existed as a recognizable topic since the mid 1980's. According to Gardiner the aim of both concurrent engineering and systems engineering is *"to provide a good product at the right time ... suitably free of*

defects and ready when the customer wants it" (Gardiner, 1996).

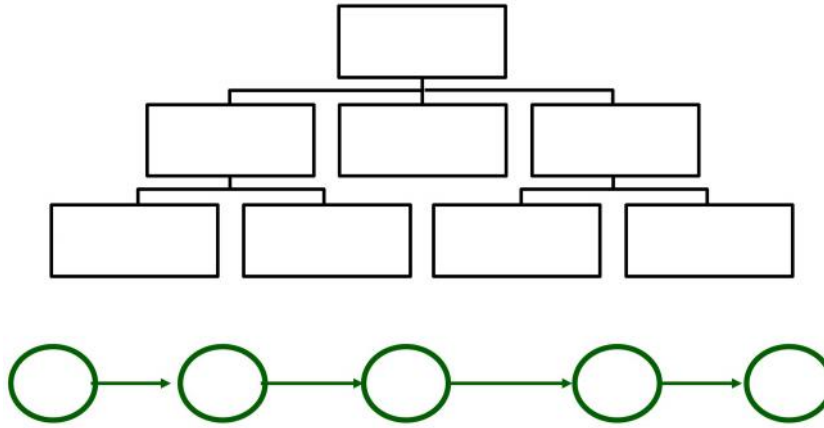


Figure 2-4 Symbology for organizations and processes

The International Organization of Standards (ISO) 8402:1994 definition of Quality is *"the totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs",* in other words 'requirements' or what the customer really wants and the degree of conformance to same'. The ISO definition of TQM is: *"management approach to an organization, centered on quality, based on the participation of all its members and aiming at long term success through customer satisfaction and benefits to all members of the organization and to society."*

Customer satisfaction is based on producing a quality product at a cost the customer is willing to pay. So, when, in early 1996, the International Council on Systems Engineering (INCOSE)[7] finally published a definition of systems engineering as *"an interdisciplinary approach and means to enable the realization of successful systems"* INCOSE seems to have reiterated the statement in the National Aeronautical and Space Administration's (NASA) manual on systems engineering, namely *"TQM is the application of systems engineering to the work environment".* This is not so surprising because many of the tools used for TQM are the same as for systems engineering, but with different names (NASA, 1992a) page 7). NASA also stated *"Statistical process control is akin to the use of technical performance and earned value measurements".*

[7] NCOSE evolved into INCOSE.

2.5 The conference

A conference is itself a system by definition. Consider the process to prepare and produce a conference for 800 systems engineers one year from today (the product). This need is analysed and a set of requirements developed for the:

- Date of the conference.
- Constraints imposed by available resources.
- Number of session tracks.
- Number of sessions.
- Anything else needed.

These requirements are analysed, alternative implementations proposed, decisions made based on evaluation criteria. A location is picked. Sessions are developed, and publicity generated. As time goes by, a call for papers is published, papers are received and evaluated, and conference registrations are received and processed. One day the conference begins and the pace really heats up as the delegates have to be provided with the full services of a conference in real-time. Several systems are in action and interacting including the preparation and presentation of the:

- Sessions
- Proceedings
- Meals
- Accommodation

Name the activities described in this section. Is it conference management or systems engineering?

2.6 The temporal perspective

Engineers were hard at work designing and building from the pyramids, war machines and irrigation systems, of pre-history to the canals and railroads of the 19th century. Those systems in their day with the technology available at the time were just as complex as the systems we design and build today. They generally had similar constraints of resources, budget and schedule. So why:

- Have the concepts of TQM only been recognized as having been around for 25-35 years?
- Has systems engineering only been recognized as a discipline since the 1950's?

The last 50 years have also seen a transition (not yet completed) from hardware-based systems to software-based systems. For example, the job advertisements in the media now tend to focus on the software skills need-

ed by applicants for systems engineering positions. Systems engineering may be an artefact of this transition.

2.7 Conclusions

Systems engineering is a discipline created to compensate for the lack of strategic technical knowledge and experience by middle and project managers in organizations functioning according to Taylor's "Principles of Scientific Management".

Most of today's systems engineers really appear (work as) to be Requirements and Interface Engineers. They have the responsibility to validate the requirements since there's little point in building a system which conforms to requirements if the requirements are incorrect. Perhaps those are two missing "ilities" in the current paradigm.

Project management, Business Process Reengineering (BPR), concurrent engineering, TQM and theoretical systems engineering all seem to be attributes of the same function; namely producing a product to (the correct) specifications by an organization within the constraints of resources, budget and schedule. Remember MIL-STD-499A was written for systems engineering management (MIL-STD-499A, 1974) and MIL-STD-499B changed the focus to systems analysis and control (MIL-STD-499B, 1992). This overlap or duplication seems to be due to defects in the current organizational structure, and in the case of systems engineering, the transition in technology from hardware to software.

We need a new organizational paradigm to simplify the organization such as the one proposed in Chapter 3 and within that paradigm, there still is a need for someone to have a strategic perspective of the entire system.

1996/7

3 There's no place for managers in a quality organization

This Chapter views the organisation from the process improvement perspective. It:

- Analyses the failures of improvement initiatives identifying three major causes of these failures, namely Quality, the structure of the organization, and middle managers.
- Discusses the need for a new organisational paradigm.
- Introduces the "*Excellence paradigm*" a paradigm for an Information Age systems and software development organization
- Presents a brief overview of the *Excellence* paradigm, how command and control is achieved without middle managers, and how work is done.
- Presents some results achieved in (and during its development), and benefits of, the new paradigm.

3.1 Root causes of the failures

The goals of a business organization are to provide a product or service needed by its customers and make the maximum amount of profit it can. Businesses in looking for ways to cut costs and increase profitability have tried various approaches such as BPR, Participative Management, IPTs, with varying degrees of success and mostly failure (Deevy, 1995) page 4). There is also growing evidence that TQM's overall success rate is so low, that for most organizations, the effort is entirely wasted (Hawley, 1995). These failures, when analysed appear to be symptoms of the following root causes: Quality, the structure of the organization, and middle managers.

3.1.1 Quality

Deming wrote "*Improvement of quality and productivity, to be successful in any company, must be a learning process, year by year, top management*

leading the whole company" (Deming, 1986) page 139). Commitment to improvement is one of the few things that cannot be delegated. The failure of top management to be perceived as being committed to Quality is a prime reason for the failure of these initiatives. In addition, there are a number of other problems with "Quality" including the following:

- **The 'Quality gap'**. Quality is taught using the terminology of the "should be" with minimal if any instruction in bridging the gap between the "as is" and "should be" states. This is because it is easier to provide information than it is to provide the wisdom to know what to do with that information.
- **Quality is not measurable**. Crosby defines quality as "*conformance to specifications*" (Crosby, 1979). Juran defines quality as "fitness for use" (Juran, 1988) page 11). The ISO 8402:1994 definition is "*the totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs*". None of these definitions prove a useful measurement of Quality. In an interview in 1991, Curt Reimann, director of the Malcolm Baldridge National Quality Award (MBNQA) said that a meaningful definition of Quality is simply not possible (Hart and Bogan, 1992) page 4). And, without a meaningful way to measure Quality, it is difficult to show top management how the benefits of improving quality affect the bottom line. For example, Contractor A can build a product to specifications for $500, and Contractor B can build the identical product to exactly the same specifications for $1000. Under the current definitions, the quality of the two identical products is the same yet the production costs are very different. No wonder Crosby wrote "*Management does not know the price of non-conformance [to quality]*" (Crosby, 1992) page 5).
- **Process improvement.** Process improvement is generally depicted as Plan Do Check Act (PDCA) and drawn as a circle as shown in Figure 3-1[8]. The use of "cycle" and "circle" imply that the organization assumes the same state periodically which leads to activity based thinking. It may be true that the improvement IPT performs each action periodically. However, the organization is in a constant state of improvement. Hence, once an improvement is incorporated, the process is different. The texts on the subject generally do not mention the need for baselines and configuration control. Consequently, the results tend to be chaotic in a large organization with several simultaneous improvement initiatives in operation. Another classic reason for the failure of process improvement initiatives is that the people involved are too busy working in the process. As A. A. Milne wrote "*Here is Edward Bear, coming down-*

[8] A better way using a spiral is discussed in Section 4.9 and shown in Figure 4-7.

stairs now, bump, bump, bump, on the back of his head behind Christopher Robin. It is, as far as he knows, the only way of coming downstairs, but sometimes he feels there really is another way, if only he could stop bumping for a moment and think of it" (Milne, 1929) page 11). An outside perspective with the time to do investigate and analyse is critical to effective process improvement.

- **ISO 9000.** A bureaucrat's dream; the process is sacrosanct! Follow it and all will be well – right – wrong! An ISO 9000 compliant process is no guarantee of Quality, only repeatable results. An organisation with a production line producing 100% rejects can be ISO 9000 compliant for as long as it stays in business.

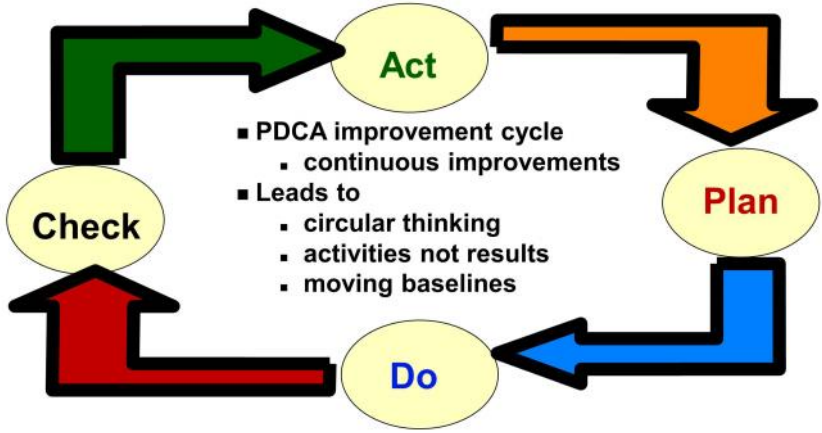


Figure 3-1 The process improvement cycle

3.1.2 The structure of the organization

The systems and software development organization is configured in a hierarchical structure. The division of work between manager and worker within our current organizational structure was discussed in Section 2.3. Later research has identified structural defects within the organization including the:

- Development of jobs which compensate for the manager's lack of skills (e.g. facilitators).
- Development of work which crosses the three nominally independent activity streams (management, development and test) shown in Figure 2-2.

In addition, project management, TQM, BPR, IPTs, concurrent engineering and systems engineering seem to be attributes of the same function, namely producing the product the customer wants.

3.1.3 Middle managers

The structure of the organisation has decoupled the decision makers from the information pertinent to the ramifications of the decision. This has produced a generation of managers typified by Scott Adams' Dilbert cartoons (Adams, 2006).

We create poor managers and then spend a lot of organizational energy (money) mitigating the effect of poor management (Chapter 2). This compensation for ineffective managers leads to excessive complexity within the organization including the creation of functional jobs such as facilitators and possibly systems engineers.

3.2 The need for a paradigm shift

Summarizing the research, the following factors point to a need for a paradigm shift:

- Excessive complexity is a symptom of an underlying problem within the foundation of the current paradigm.
- The many failures of the current adaptive approaches to improving our organizations.
- The report producing and information filtering functions of middle management have largely been replaced by technology (Rodgers, et al., 1993).
- The adoption of project management and other sub-organizations which cross the boundaries of the three streams of work (e.g. IPTs and concurrent engineering). Our symbology uses boxes for a hierarchical organization structure and circles for a process as shown in Figure 2-4. Truly a case of attempting to insert a square peg into a round hole.
- Management consulting has become a major growth industry grossing over $7 billion each year (Deevy, 1995) page 25). Engineers are expected to know how to engineer; physicians are expected to know medicine, yet we don't expect managers to know how to manage!
- Middle managers don't seem to be adopting the recent spate of "new management" ideas. This is not surprising since it is nigh impossible for people to 'unlearn' what they know is the correct way to do something (Kuhn, 1970). The failure of BPR could have been predicted just by looking at the unfortunate choice of words on the cover of the book that introduced BPR, i.e. "*Forget what you know*" and "*most of it is wrong*" (Hammer and Champy, 1993) which violate Kuhn's rule.

3.3 The Excellence paradigm

Management is considered as the planning, organizing, directing and controlling a time ordered sequence of activities (process) performed by people

building a product. (Harrington, 1995) page 1), stated **"Stop worrying about quality, productivity, cost, and cycle time. Focus your energies on organizational performance improvement and all the rest will follow".** The *Excellence* paradigm takes that advice and eliminates the root cause of organizational problems by being based on a vision of an organization performing the functions of middle management but without managers. It is thus a real application of the Reengineering idea (Hammer and Champy, 1993) because while the basic approach is similar to Taylor's approach (Taylor, 1911), it is not based on his assumptions. The *Excellence* paradigm:

- Uses technology to replace many functions currently performed by managers.
- Provides a model that may lead to a theoretical basis for understanding why the excellent organizations reported in Peters and Waterman are successful (Peters and Waterman, 1982).
- Moves decisions to where the action is taking place.
- Integrates a set of elements in a holistic manner. These elements outlined below are the:
    - Structure of the organization.
    - Corporate culture
    - Quality-Index
    - Division of Labour
    - Effective task management
    - Event/product based cost accounting
    - Dynamic organization
    - Effective use of technology
    - Reward and recognition system

    The *Excellence* paradigm, in its development phases, has:

- Reduced the cost of work on various short duration projects by a factor of 10^9.
- Enabled the design and development of a network of 600 microprocessors controlling the LuZ SEGS-1 solar-fuelled electrical power generating system, such that it was installed half way around the world and worked first time with only a single hardware discrepancy report[10].
- Saved NASA's Goddard Space Flight Center $1.5 million (Kasser, 2013) pages 383-385).

Consider each of these elements.

---

[9] See Chapter 22.

[10] See Sections 11.8.1 and 18.9.

3.3.1 The structure of the organization

The *Excellence* paradigm is process-based (Hammer and Champy, 1993) page 28), uses a systems approach and considers an organization as a four dimensional system (product, process, people and time) as shown in Figure 3-2. As such, systems engineering methodologies are used to view, decompose and optimise the organization.

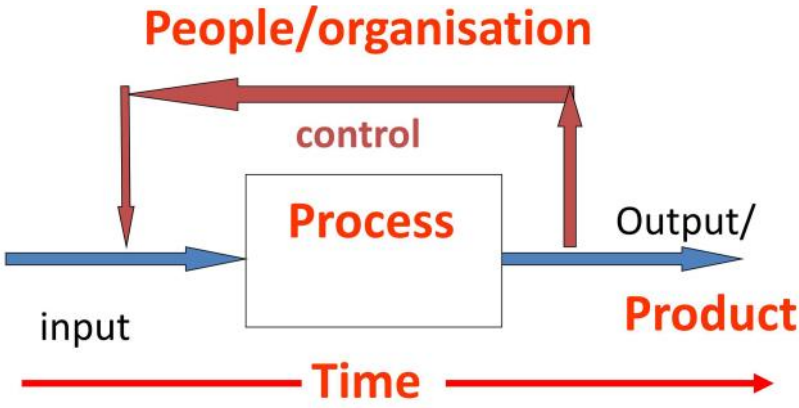


Figure 3-2 The PPPT model

The environmental model, for example, shows the organization exists in a marketplace comprising suppliers and buyers. In the buyer marketplace, customers are a subset of the potential market. The cost of finding new customers is greater than the cost of servicing existing customers, so companies tend to focus on retaining existing customers. This is all right; however, customers are part of the environment. They are a subset of the population as a whole. However, companies can go out of business by:

- Concentrating on satisfying existing customers without trying to attract new ones
- The lack of an awareness of changing technologies.

For example, carburettor manufacturers may have had an excellent product, and satisfied customers, but fuel-injectors eliminated their market.

Existing customers also tend to recommend adaptive improvements to products. They may not know they need innovative ones, and only recognize them when they appear. For example, customers did not demand telephones, electric lights, automobiles, Sony's Walkman, fuel injectors, etc. until they had been invented and marketed.

Consider the organization as two major systems, namely, a:

- **Production system** which produces the product from which the organization makes its profits. Anything in this system is a direct charge.
- **Support system** which provides the support to the production system (including control and feedback).

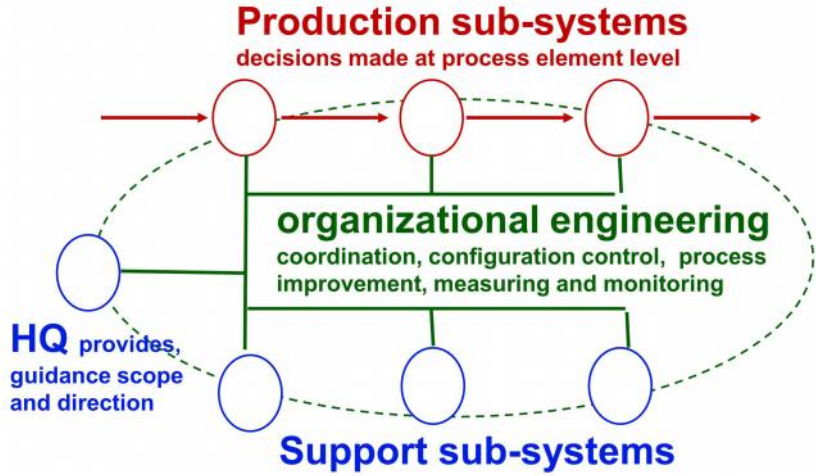


Figure 3-3 Organisational engineering in the *Excellence* organization

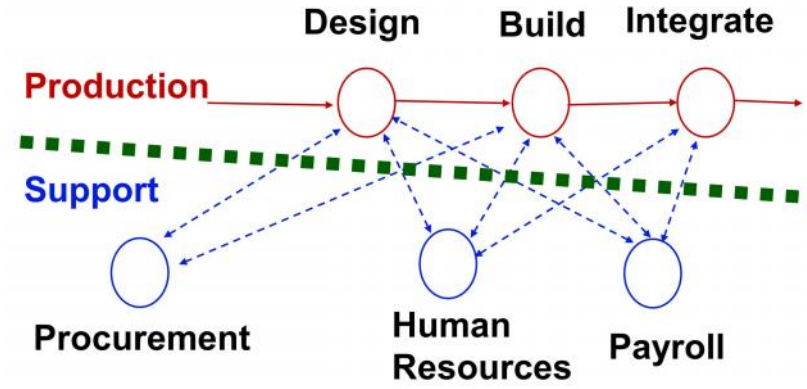


Figure 3-4 The *Excellence* organization paradigm

- There is no single view representation of the structure of the organisation. However, various views are discussed in this book and two views are shown in Figure 3-3 and Figure 3-4. The term organizational engineering has been introduced to emphasise that the focus is on:
  - organising the people within the organisation;
  - organising the processes in which the people are active;
  - organising the products produced by the people;

- organising the adoption of new technology as and when appropriate;
- organising the information system that makes everything else possible;
- organising the organisation in which everything else exists.

This organising is performed in the manner of engineering any other type of system. Engineers ran organisations before managers, for example Frederick W. Taylor presented his paper on "Shop Management" to a meeting of the American Society of Mechanical Engineers (George, 1972) page 92) and other early 20th century leaders of management thought such as Harrington Emerson and Henry Gantt published in engineering journals (George, 1972) pages 104-107).

Information and products flow between the organisational work elements which can be shown in a process flow or Program Evaluation and Review Technique (PERT) chart. Decisions are made at the appropriate level to maximize cohesion and minimize coupling between work elements. Each bubble in the chart may consist of:

- Both process and support elements.
- **Traditional organization elements** - managers and workers, as not everyone wants to be, or is ready to be, empowered.
- **Self-directed teams** - IPTs empowered by the leadership of the organization.

The shift from traditional management to self-directed teams can take place in a gradual manner, suitably reinforced by the organization's RRS. Each process circle has both process and support elements (functions). The analogy of these teams to the computer world is:

- **Hardware: printed circuits** - contain signal processing (production) and power conditioning (support) circuitry.
- **Software: objects** - contain programs and data.
- **Systems: systems** - contain production and support elements.

3.3.2 Perspectives on the process-organization

The process perspective maps into Covey's circles of influence and circles of concern and provides a new perspective with which to view the organization (Covey, 1989). For example, Figure 3-5 shows a section of a buyer-supplier chain in an industry. The view is of stages in the production process. Organizational boundaries are shown as dotted lines. Organization AB performs two stages in sequence and sells the product to the organization which performs process C. Organization EZ produces two products which it sells to organization/process W. If organisations EZ or AB are seeking to expand, a logical merger or acquisition is with organization/process C. By acquiring C

organisation AB integrates the supply chain, while by acquiring C organisation EZ establishes a monopoly in the supply chain to W. Notice that organization OX produces two products, selling to different customers and obtains raw materials from two different suppliers. Organisation OX needs to review its mission. By viewing the circles from the perspective of the entire process, or the functions serving a specific customer, teaming opportunities show up. For example, in the Government contracting arena, this view identifies opportunities for forming strategic alliances on a functional split for bidding in response to requests for proposals.

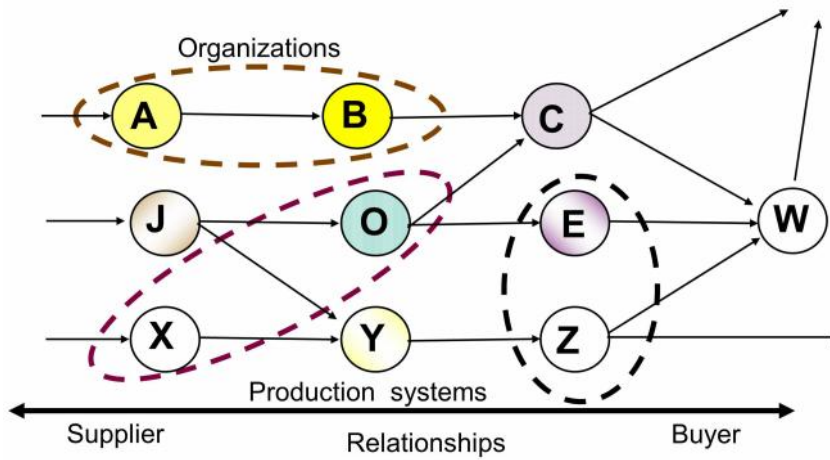


Figure 3-5 The buyer-supplier perspective

In today's paradigm, much is made of the difference between products and services, and the intangibles involved in services that tend to prohibit them from being measured. Yet someone or something produces the service. Therefore at some point in the supply chain, a service is a product as shown in Figure 3-6.

The process-based organisation must not be considered from the perspective of the activities performed in the process, but instead it must be considered from the perspective of the products produced by each process element. This perspective means that:

- The organisational process elements can be considered as "black boxes" with well-defined inputs, outputs and transfer functions.
- The organisational entities can be well-designed minimizing coupling between elements (Hammer and Champy, 1993) page 123).
- It is now possible to allocate authority and responsibility for the entire process producing a product. This can correct the findings of Hammer and Champy who state *"in most companies today, no one is in charge of the processes*" (Hammer and Champy, 1993) page 28).

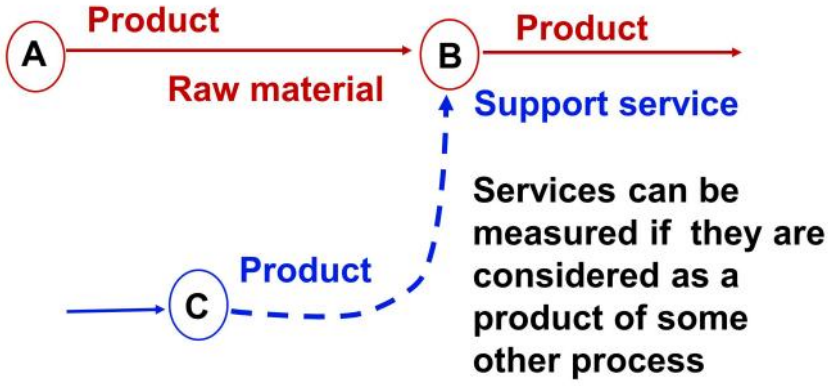


Figure 3-6 Products and services

- Shows that "*the legal entity, the company, is a reality for shareholders, for creditors, for employees and for tax collectors. But economically it is fiction*" (Drucker, 1995) page 126). In other words it is an area of interest within a boundary enclosing a section of the supply chain delivering a product to a customer.
- Cost accounts can track the cost to produce products, rather than the cost of doing work.

3.3.3 Corporate culture

The corporate culture is based on the three principles of Completeness (Crosby, 1992) page 19) namely:

1. Cause employees to be successful.
2. Cause suppliers to be successful.
3. Cause customers to be successful.

This is the underlying basis for *Excellence.* How does the organisation cause employees, suppliers and customers to be successful? It does that by establishing an environment that allows effective people to do their thing with a minimal amount of interference. In my first job as a manager I realised that one of the most important things I had to do was shield the engineers and technicians form the organisational bureaucratic crap and let them get on with their work. What a waste of my time, and what a squandering of resources paying the bureaucrats to produce the crap in the first place. The *Excellence* organisation avoids the Peter principle (Peter and Hull, 1969) by promoting effective people, not as a reward for doing an outstanding job, but on their ability to do the job into which they are promoted. As a person moves higher in the organisation, the skills needed to do the job change and in many organisations by being rewarded for doing an outstanding job, people tend to be promoted to one level above their competence a

situation which has been termed the Peter Principle. This situation is illustrated in Figure 2-3 showing the promotion from a technical to a managerial position. A person should be rewarded for doing an outstanding job, but not by a promotion. Other means exist and are effective for example, a cash bonus is used in industry and medals are employed in the military. The organisation's RRS should provide the means of rewarding performance. The RRS is also associated with career planning to grow employee's skills as well as education and training. Education provides a promotion path, while training leads to excellence in the performance of the current task.

- **Training** is provided to increase skills and competences in performing tasks. Thus training improves how a task is performed.
- **Education** is provided to increase insight and understanding why the task is performed (Hammer and Champy, 1993) page 71). However, education must be relevant to the business (Gouillart and Kelly, 1995) page 274). Once an employee has received a postgraduate degree or other educational qualification, they should be allowed to use their knowledge. When I was at University of Maryland University College (UMUC) several of my students who qualified for their Master's degree moved to new organisations because the employer who paid for the education did not allow them to make use of their newly acquired skills and knowledge. What a waste of money and talent!

The purpose of the *Excellence* organisation is to create wealth by providing value to its customers and making a fair profit. The organisation aims to be the best in its field. The organisation views employees as partners with a stake in the future of the organisation (Ford and Crowther, 1922) consequently the employer as a leader in its industry should try pay above average salaries, while the employees should make this possible by virtue of their productivity (Ford and Crowther, 1922) page 117). To make this happen, the culture of the organisation is that of learning. There is no such thing as a learning organisation. An organisation can foster a learning culture and facilitate and reward learning, but it is the people within the organisation who do the learning[11]. Learning must be continuous, since current knowledge becomes obsolescent within a few years. However, the learning should take place in the correct environment. Drucker wrote, "*Many big companies are currently building their own in-house education facilities. I advise caution here. The greatest danger to the big company is the belief that there is a right way, a wrong way, and our way. In-house training tends to emphasize and strengthen that view. Skills, yes; teach them in house. But for purposes of broadening the horizon, questioning established beliefs, and for organized*

[11] And take that knowledge away with them when they leave.

abandonment, it is better to be confronted with diversity and challenge. For these managers should be exposed to people who work for different companies and do things in different ways" (Drucker, 1993) page 350). Tertiary education courses in Information Technology (IT), systems and software engineering need to be updated each time they run to make the readings current. Sometimes only minor updates are needed, at other times major updates are required. For example, at the beginning of 2000, any course that contained material on the Y2K problem had to be revised. Thus in light of the current shortage of qualified personnel, courses may not get updated in a timely manner, and consequently the education provided to the students is less than optimal. Some universities are already solving part of the dilemma by purchasing courses that they do not have the resources to produce on their own. The World Wide Web has, for the first time, made possible a significant change to historical paradigm of the university as a central learning location (Kasser, 2000b). Investigate and select the correct tertiary institution for your organisation based on course content as well as flexible delivery mode.

Each element of the organization is self-regulating within the boundaries set by the culture. Each element has a vision of what it is supposed to do, the resources to do the job and the schedule. Command and control is based on:

- Management by objectives.
- Costing products not activities.
- Ensuring products are handed over from one stage on the process to the next, namely there is a transaction at the process interface.
- Ensuring resources are available when needed.
- Making decisions at the working level.
- Using technology to:
    - Communicate formally and informally.
    - Monitor progress is within planned limits, using the principles of management by exception and statistical process control (SPC).

3.3.4 The Quality-Index

Deming wrote "*Quality comes not from inspection, but from improvement of the production process*" (Deming, 1986) page 29). He also wrote "*Defects are not free. Somebody makes them, and gets paid for making them*" (Deming, 1986) page 11). The product is produced by a process within an organizational environment. As such, the process, product and organization represent three tightly coupled dimensions of quality and must not be considered independently (Kasser, 1995). So, to use this concept, define a Quality-Index along the lines of the MBNQA (Kasser, 1995), where the Quality-Index of an organization is a three dimensional measure of the:

- Degree of conformance of the product to its requirements, namely the definition of Quality (Crosby, 1979).
- Effectiveness of the production process.
- Effectiveness of the organization in which the process takes place.

This use of the Quality-Index means, for example:

- The effect of poor management may be defined in terms of "cost escalations" instead of using soft "Quality" based language. This means we can now talk to top management in terms of "cost reductions" which they understand (the customer's language), rather than "quality" which they generally don't.
- Activities that lower the cost of 'producing the product within specifications' improve its Quality-Index. These activities take place in the area marked as Organizational Engineering as shown in Figure 3-3.
- In the example above (Section 3.1.1), the process dimension of the Quality-Index of Contractor A is at twice the value of Contractor B.

3.3.5 Division of labour in the Excellence organization

The division of labour is between strategic and tactical as shown in Figure 3-7 and split as:

- **Strategic.** The strategic planning, coordinating and "communicating the vision" functions.
- **Tactical.** Production work, measuring, self-directing, tactical planning; the activities within a process element.

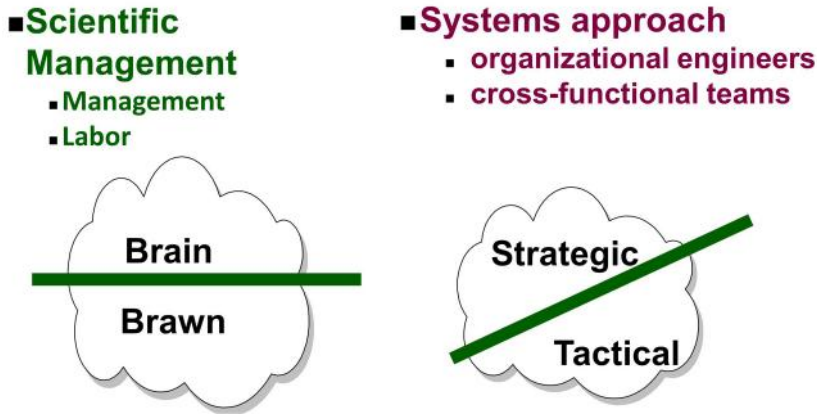


Figure 3-7 Division of labour

The difference between strategic and tactical depends on how the work is viewed. For example, an admiral-of-the-fleet performs strategic functions, while each ship's captain performs tactical functions. Yet, within a ship, the

captain performs strategic functions and the engine room rating performs tactical functions.

This division of work provides a true dual promotion path. Promotion from process IPT to process improvement IPT is a strategic promotion path for those wishing to assume more responsibility within the organization yet remain in a technical position. Technical specialists, who can mentor junior personnel, serve as part time members of, and advisors and consultants to, the IPTs.

3.3.6 Effective task management and anticipatory testing

Work in the *Excellence* organization focuses on events, products and results not on activities. "*Prevention is planned anticipation*" (Crosby, 1981). Organizational engineers use 'communicating the vision', management by exception, management by objectives, prevention of defects, testing, and developing and using metrics to maximize the Quality-Index to monitor and control the work. As a result work tends to be done the right way the first time, so the cost of the process is reduced. All tasks are visible in a Network Scheduling Tool (NST) displayed in the manner of a process flow or PERT chart. The description of the tasks performed in a process map directly to the job descriptions of the people in the tasks. This eases writing job descriptions and requirements for people to perform them. Any task that is not producing something measurable is questionable. The people performing each task know their customers are in the next task bubble in the chart.

3.3.7 Event/product driven cost accounting

The Work Breakdown Structure (WBS) represents the allocation of work elements and cost accounts as follows:

- Work elements to the task based on the products to be produced for specific events or milestones by the task.
- Cost accounts to the WBS elements. In this way, the cost of:
    - The work performed to comply with a task requirement is recorded and can be used as a baseline to refine future cost estimates.
    - Each specific product or event is known.

The traditional approach is to use a WBS to allocate work by listing the activities in the hierarchical format of the WBS. This approach often allows some work elements (processes) to be overlooked. The Event/Product driven cost accounting approach recognises that a WBS is a hierarchical representation of the work being done, namely of a process, and uses a process-based approach to develop the activities and resources as follows. The basic tool is the Product-Activity-Milestone (PAM) chart (Kasser, 1995) shown in Figure 3-8. This tool has been found to be very useful in developing the rela-

tionships between the product, the activities and the milestone. The PAM chart is similar to, a cause-and-effect chart and is a tool to facilitate:

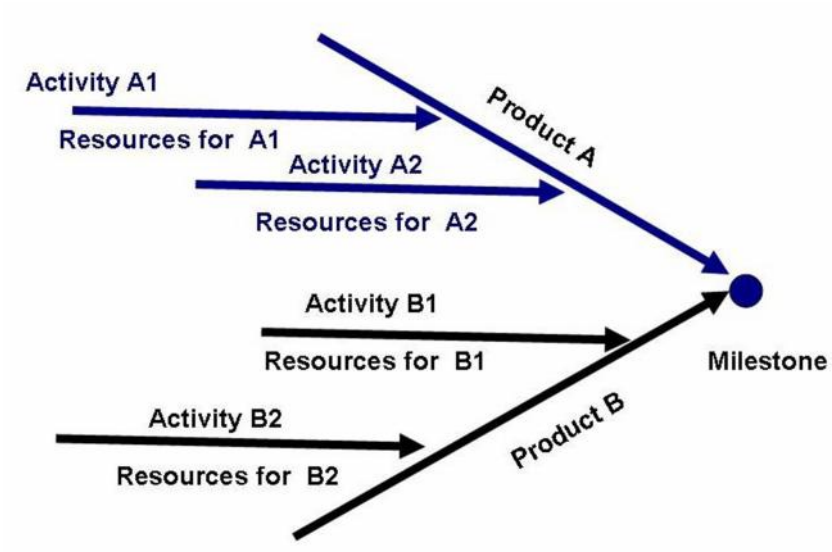


Figure 3-8 The Product-Activity-Milestone chart (Kasser, 1995)

- Defining a point in time (milestone).
- Defining the product(s) or goals to be achieved by the milestone
- Determining the activities to produce the product(s).
- Defining the resources needed to produce the product(s).

The PAM chart consists of four parts:

- **The milestone** - Shown as a circle.
- **The product(s) produced** - Drawn as a sloping line(s) leading towards the milestone. Two products (A and B) are shown in the Figure.
- **The activities** - Drawn as horizontal lines leading to the product line. They are listed above the line. Labelling reflects the activities associated with the product, so activities A1 and A2 are associated with Product A, and activities B1 and B2 are associated with product B.
- **The resources associated with each activity** - Shown as labels below the activity lines. They are listed below the line. Labelling reflects the resource associated with the activities, so resources for A1 are listed below A1, resources for A2 are listed below A2 etc.

The PAM Chart is implemented using paper and pencil. Starting with a blank page, a milestone is positioned at the end of the paper. Arrows are drawn on the product and activity lines to show the direction of progress. Note there may be more than one milestone within the chart. This is be-

cause the simple PAM chart does not explicitly show any activities and resources needed to integrate the products for the milestone.

Although the labels have used letters in this example, in practice you would use WBS style numerical listings. Thus for example, Product 3 would have WBS elements 3.1, 3.2 etc. Once the PAM chart for the products to be produced for a specific milestone has been developed, you use a PAM chart to assign:

- **Work elements to the task** by linking them to complying with requirements. In this way, the cost of the work performed to comply with a task requirement will be recorded and can be used as a baseline to refine future cost estimates.
- **Cost accounts to the WBS** elements on the basis of the products to be produced by the task. This approach will allow the customer to know exactly how much a specific product produced by a task actually cost.

This approach ensures that the WBS accurately maps onto all the work in the project providing a 100% mapping of the WBS onto the Project Breakdown Structure (PBS), namely there is never a difference between the WBS and the PBS. The focus of the WBS is on results (products and events). Product based costing does not only apply to products produced for sale. It also applies to products used within the organization (indirect products) as shown by the following example. The real cost of replacing employees in specific positions or filling new positions tends to be unknown in many companies. It can readily be determined using product based costing in the following manner. Consider the process of filling a position as a mini project in which the completion milestone is the employee performance review which generally takes place 90 days after an employee has been hired for the position. The product is a satisfactory employee. The requirements for the product are the skills, education, excellence and experience needed by the candidate to perform the job. The activities to arrive at the milestone include:

- Drafting the employment advertisement.
- Placing the advertisement.
- Reading the resumes received in response to the advertisement.
- Deciding which candidates to interview.
- Interviewing the candidates.
- Making the hiring decision.
- Hiring the best candidate.
- Orienting the candidate.
- Any new employee training.
- The 90-day performance review.

Charge all products and activities to the project account. If any products or activities are shared amongst a number of job positions (i.e. employment advertisements), then pro-rate the charges accordingly. At the end of the time, you will know what it cost to fill the position. Once you have this information for several hiring instances, you will know:

- The cost avoidance of reducing employee turnover.
- The total cost of the hiring process. This may then be broken out to identify which parts of the process could be improved (reduced in cost).

3.3.8 The dynamic organization

In the *Excellence* paradigm, process improvement:

- **Is continuous.** The organization is in a state of dynamic equilibrium. Improvement of quality and productivity, to be successful in any company, must be a learning process, year by year, top management leading the whole company (Deming, 1986).
- **Is a process in itself and needs to be compliant to standards.** Conceptually, upgrading a process is no different from upgrading a product release. Both upgrade a system and must be performed in an appropriate manner (change requests (improvement suggestions), impact assessments and configuration control). Approved changes are then carried out at specific milestones. There is no excuse for chaos while implementing changes.
- **Must be performed by a separate IPT of people working interdependently with the team who perform the process**[12]**.** While the people involved in the process measure the process (Peters, 1987) page 90), process improvement requires a different set of skills to working in the process. People working in a process are generally too busy to take the time to improve it properly and objectively. Outsiders are more likely to be objective and challenge the assumptions inherent in a situation.
- **Is focussed on improving the organisation not on improving parts.** Focus on improving parts results in situations similar to the garden gnome example discussed in Section 4.1. The process improvement team (PIT) is an IPT which gets full disclosure and suggestions for improvement from the people performing the process, then holistically analyses the information and suggestions from the system's perspective to determine the effect of the proposed improvement on all parts of the process.
- **Is cost-effective.** The cost of the improvement initiative has to be less that the cost reduction achieved.

[12] See Chapter 19.

- **Results in reorganisations as process elements are changed.** This may be because production paths are rerouted, projects and production runs end, or new lines are started up. The organizational engineering function must ensure that the process organisation does not suffer from horizontal (process) stove-piping in the manner of the vertical stove-piping of hierarchical organisations. In the process organisation the effect of reorganisations can be seen in the new process flow diagram and the products passed across the process interfaces.

3.3.9 Effective use of technology

Today's organizations exist in an environment that has become recognized as being complex. The future of organizations depend on the making of informed decisions at all five layers of systems engineering, from decisions at the strategic layer in the Boardroom, to decisions that affect the production process at the lowest layer. Making these optimal decisions requires that correct and current pertinent information be available when the decision is to be made. According to Farnham, the problem the executive had was to secure at all times, live and accurate data concerning the exact conditions of the business (Farnham, 1920) page 20). Yet, in the subsequent 85 years we have not developed an accounting system which tells the decision makers what their costs really are or a management information system that provides pertinent information for making an informed decision between two alternative courses of action. Consider what such an information system would do and what form it would take.

3.3.9.1 What the information system would do.

Beer provides a description of the conceptual information system (Beer, 1972) page 244). He wrote that bits and pieces of it existed yet in the intervening years, although the technology to build such a system has become commonplace, it still does not exist, at least in the literature. Beer discusses the British War Room in the Battle of Britain as a close parallel, and NASA's control room at the Manned Space Flight Center in Houston, Texas[13] as another example. Information pertaining to the process flows in the organisation, the resources available and schedules would be accessible. This information exists in digital form in most organisations; it is just not readily accessible in a manner to assist the decision makers. The information system would allow decision makers to:

[13] While Beer proposed a control centre in a room, today's technology allows for personal desktop portals accessing information via software agents in the manner of the FREDIE described in Chapter 8.

- Make "what if" analysis of decision options would also be available, some being provided from external sources such as Internet Search Engines, and Databases.
- Run simulations of the effects of alternative decisions in faster than real time to see how the decision is sensitive to various assumptions.
- Perform safety analyses to determine how the system will react to hazards.
- Access all information available to their clearance level. Today's organisations suffer and governments fall as a result of a persons in a position of responsibility misinforming those they report to, or hiding information and problems until it is too late to take corrective action. This element of the information management system will minimise that capability.

3.3.9.2 The form it might take

It might be an assistant to the decision maker and a tool for the analyst. It could be called PERCY which stands for **PER**sonal portal into **CY**berspace. PERCY is a portal into an organisational information system architecture which maps onto the organisation's processes. The architecture is distributed over the organisation's local and wide area networks using local databases, not a centralised computer[14]. Each process element inputs data into the system and makes use of data from the system. The information flows between the elements of the system is arranged for control and status data complying with an Interface Standard. This means that the system can be assembled from current databases and implemented slowly, connecting one element at a time using legacy databases. Access to the raw data is via software agents in the PERCY that construct a query and determine the location of the wanted data and send a request on the network for the data. When the reply is received, the software agent formats the information and displays it in the appropriate manner in the PERCY. PERCY will thus operate in a network centric or Integrated Digital Environment (IDE) that contains

- local databases on the person's desk,
- remote legacy and yet to be developed databases on the corporate local area network,
- external data on the World Wide Web,
- data in non-electronic formats (e.g. books, drawings, and journals).

A PERCY will have to provide timely access to this information but will also have to process the information and provide reports in the user's language. This environment and functionality can be considered as complex by

[14] While a distributed architecture poses some design difficulties, the central computer architecture is vulnerable to many threats.

all definitions of the term. It is not practical to build a complex knowledge management system at one time rather an incremental approach will be needed. Kemp et al. write that the knowledge management community needs to address several essential issues immediately (Kemp, et al., 2001), including:

- **A systems approach.** Typically, knowledge management programs focus on narrow solutions. A holistic approach is needed.
- **An evolutionary process.** There is no currently accepted process model that supports the continued evolution of knowledge management capabilities within the organisation.

The PERCY-IDE or network centric approach to dealing with the complexity of this situation is not to use the traditional systems engineering development methodology that gathers the entire set of requirements up front before commencing the project. Instead it should be built out of legacy entities that have various formats and ways of storage. The PERCY-IDE project allows the system to evolve within a broad architecture framework using the Rapid Incremental Solution Construction (Kasser and Cook, 2003) approach[15]. The approach is reductionist, namely to first provide a solution for a part of the problem, and then provide a solution for another part, and so on, until the collaborative system that includes encapsulated tools, newly designed custom software and human collaborators (Lander, 1997) is developed. There will be many data sources in the PERCY-IDE including:

- Internet Web sites (URLs)
- Databases (free and subscribed, local and remote)
- CD-ROMS (personal and available upon request)
- Libraries (personal and available upon request)
- Printed media (journals, books, notes)
- Project related information from Corporate sources
- Other Corporate data sources.
- Its own knowledge base – needed to get a base line into a problem

The PERCY is a tool that performs activity on information. However, the types of activities performed on information in the various layers of an organization are numerous. Hence the PERCY can be expected to take various forms including

- Corporate portals;
- Executive decision making portals;
- Domain expertise portals;

[15] And the Cataract Methodology described in Chapter 13.

- Others.

Technology has many uses, but must be applied in an integrated manner. Thus, information in a computer in one part of the organization, or in one software application must be accessible by others. This requires interoperability and compliance to Industry Standards, not to manufacturers' standards. Moreover, incompatibility can be present even within the product range of one manufacturer. Two such examples in word processing were WordPerfect 5.1, which couldn't read WordPerfect 6 format files, and Microsoft Office 2007 files which could not be read by earlier versions of the product until the file format converter was released. This lack of compatibility in one producer's products requires people to upgrade just to be able to exchange data files, even if the additional features in the upgraded software are not needed or desired. There is also the problem of exchanging files between programs produced by different manufacturers (e.g., Word and WordPerfect).

Technology must be used in a seamless and almost invisible manner. For example, information should be entered into a computer only once. Thus data entry forms should be front ends into databases and used for data entry and retrieval. However, technology should only be used where it is cost effective. A flow chart template, and paper and pencil may still be more cost effective than software in documenting processes in your situation.

3.3.10 Reward and recognition system

An organisation's management systems – the ways in which people are paid, the measures by which their performance is evaluated, and so forth – are the primary shapers of employees' values and beliefs" (Hammer and Champy, 1993) page 75). Winning (world-class) organisations need to focus on individual excellence and reward individuals for their achievements and the risks that they are willing to take (Harrington, 2000). Yet performance evaluations are discouraged (Deming, 1986) for many reasons including the following:

- **Measurements are subjective.** The argument is that subjective measurements demoralise people so don't bother to make such measurements. If indeed measurements are subjective, then the search should be started for objective measurements.
- **Measurements are made based on arbitrary goals.** The argument is that since the goals are arbitrary they may not be achievable or desired by the employee. The goals should be set in a participative manner with the employee contributing, understanding the need for, and taking ownership, of the goals. They will then cease to be arbitrary.
- **The system is at fault and people's performance cannot improve within the boundaries of the system.** Deming's "Red Bead Experiment" is often quoted to reinforce this interpretation (Deming, 1986). However

Deming's comments about changing the system have been conveniently forgotten. The *Excellence* paradigm has changed the system.

- **Half the people will always be performing below average.** This argument has been used as a shield for poor performance. The fallacy in the argument is the definition of average. The systems perspective (environmental model) is to define the average as the average for the industry not the average of the individuals in the specific organization. Thus, the idea is to position the organization as far above the industry average of organizations as possible as shown in Figure 3-9.

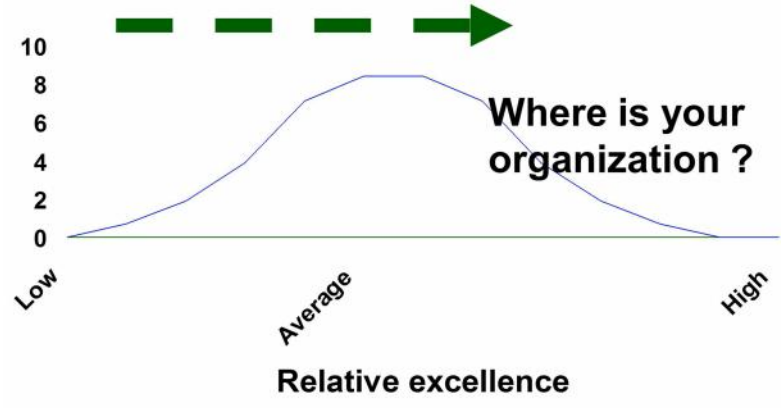


Figure 3-9 Comparative position of organisations within Industry

So the problem is not the performance evaluation, it is in the way it is implemented. According to Henry Ford "*If an employer urges men to do their best, and the men learn after a while that their best does not bring any reward, then they naturally drop back into "getting by"* (Ford and Crowther, 1922) page 117). A Reward and Recognition system (RRS) is a people-oriented system as shown in Figure 3-10. It is bi-directional and performed by people on people, using processes and evaluation criteria created by people. Consequently, an understanding of people is critical to the success of the RRS.

Many ways of categorizing people have been developed over the years. Consider the use of the classification developed in the book Guerrilla Selling which was chosen because the classifications are made in the way people relate to each other (Gallagher, et al., 1992). Gallagher et al. use the context of seller to prospective buyer and discuss the following types of personality:

- **Externals** - Who work based on inputs, statistics and testimonials. These types of people need regular praise, recognition and feedback.

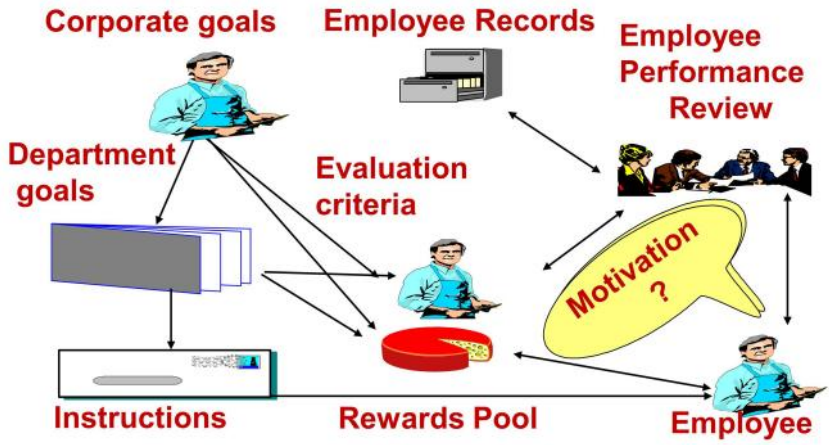


Figure 3-10 Elements of a reward and recognition system

- **Internals** - Who work based on opinions, feelings and values. These types of people let actions speak for themselves, are not interested in recognition and awards, but work to meet their internal standards and goals.

The organization's RRS must reinforce behaviour that is in accordance with the values of the organization (Harrington, 1995) page 469) for both types of personality. People's behaviour is explained by several theories including McGregor who postulates two opposing types of behaviour (Theory X and Theory Y) (McGregor, 1960). From the perspective of systems thinking, the difficulty in reconciling the two types of behaviour may be because they are not so much opposing, but they are two ends of a situational continuum. As a result the same person can exhibit Theory X behaviour in one situation, and Theory Y behaviour in another. My children provided a perfect example of this continuum when they were young. They needed no motivating for some tasks (e.g. eating ice cream), while other tasks required an enormous amount of parental energy expenditure to get the children to perform them (e.g. cleaning up their rooms). The aim of the RRS is to gently move employees toward the Theory Y end of the continuum. The keys to developing an effective performance process are (Harrington, 2000):

- Measure the right things or select the right evaluation criteria.
- The employee and manager agree to the performance standards.
- On-going measurement and feedback.
- Formal evaluations should be conducted when a milestone is achieved, not according to an arbitrary calendar date.

The evaluation criteria you choose to achieve this purpose are critical. Consider using some of the following:

- **The seven habits of highly effective people** (Covey, 1989).
- **Individual contributions to their project** - Based on the contribution of each member of the team to the development of the product and the improvement of the process. Both attributes may be measured by managers and peers.
- **Team spirit** - Based on how each member of the team works together with the team and contributes to the success of the team. These attributes are measured by peers.
- **Contribution to company growth and reputation** - Based on volunteer work on proposals, adopted suggestions for process improvement in areas within and outside the person's work area; how they grow and improve other people in the project; letters of commendation and awards from customers and other sources external to the organization.
- **Personal growth –** Based on demonstrated willingness to learn and develop (Hammer and Champy, 1993) page 189), evidenced by courses taken, conferences attended, technical journal articles published, and conference papers presented.
- **People skills** - Negotiation skills and other such skills for working with people (Lewicki and Litterer, 1985).

Reward and recognition is an on-going process. Ways of identifying rewards and delivering them were discussed in "Applying TQM to Systems Engineering" (Kasser, 1995) pages 18-19). Evaluation of personnel takes place at appropriate times. The evaluation criteria must be posted and known to all employees. Never evaluate an employee against the "broader picture" or undocumented criteria. Once a manager has evaluated an employee against unspecified criteria, even if the employee doesn't file a complaint, that employee will no longer trust the manager, because even if mutually agreed goals are subsequently set, there is no guarantee that the manager will not repeat the process at the next performance evaluation and evaluate the employee against another set of unspecified criteria.

An evaluation is made on each of the criteria. The reason for the evaluation against each criterion is documented (also important for legal and regulatory compliance reasons). The grading of the employee with respect to the evaluation criteria must be objective and fair. Evaluation may be made by several different people and the results for any specific criterion are a weighted sum of all the evaluations. The evaluations for each criterion (if performed by different people) and the normalized results may be plotted for each employee as a bar chart as shown in Figure 3-11. Each criterion also has upper and lower limits just like a Statistical Process Control chart. The upper and lower limits are set so that normal behaviour is within the limits.

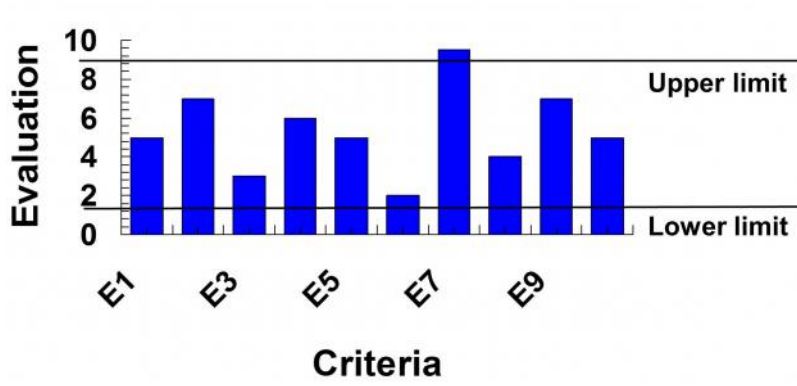


Figure 3-11 Performance evaluation chart

If the RRS is working correctly, most evaluations should fall within the upper and lower limits showing that the process is in control. Any situation in which an employee receives evaluations outside the limits is to be investigated by an independent organisational element to ensure the evaluation was fair and determine the reason for the exceeding of the limit. Exceeding the upper limit may show excellence or favouritism as well as outstanding performance; falling below the lower limit shows something entirely different. Each has to be investigated. Thus in the figure, the employee has exceeded the upper limit for Evaluation Criterion E7, and the reason needs to be determined.

The evaluations also have to be checked over time to learn if there is an abnormal pattern. For example there may be a supervisor who never gives a certain employee a good evaluation. The mediocre evaluation may be out of phase with other elements of the evaluation at that time or with the employee's performance history. In today's litigious society these checks are becoming important to ensure that the evaluations reflect the real performance. Today's technology can perform "pattern checking" on evaluations to weed out this situation. A typical performance evaluation chart for two time periods for a different employee is shown in Figure 3-12. Note the changes in the second evaluation showing an improvement in some of the evaluation criteria.

In time, as the *Excellence* paradigm is adopted the industry average of organizational effectiveness will tend to move up the continuum towards maximum effectiveness. This effect is seen today in the product dimension. For example, as noted and then predicted by Moore's law[16], the semicon-

[16] Gordon E. Moore predicted that the number of transistors the industry would be able to place on an integrated circuit wafer would double every year. He updated his prediction to once every two years in 1975.

ductor industry has quadrupled the density of random access memory integrated circuits every few years.

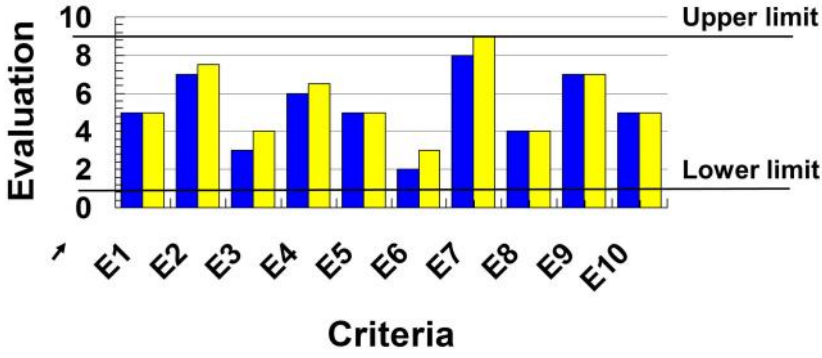


Figure 3-12 Performance evaluation chart for two time periods

3.4 Summary

This Chapter summarized an analysis of the defects in our current organization situation and provided a brief overview of a new cost effective paradigm.

1997

4 Systems engineering the *Excellence* organization

The hardest part of making the transition to the *Excellence* (customer-based) organization is the first few steps and not only need it not be chaotic it must be orderly. Thus from the perspective of process improvement, this Chapter describes the first few steps in the transition process, showing how to:

- Identify internal and external customers.
- Identify metrics to measure performance within the organization.
- Use transactions to ease the path to the new paradigm.
- Use transactions to minimize the resistance to change.
- View an organization as having two sets of internal customers, namely:
    - **Vertical** - the traditional reporting path in which the customer is the supervisor.
    - **Horizontal** - the production process path in which the customer is the next link in the production process. This customer may be internal or external.
- Set up evaluation criteria to measure the effectiveness of the person in dealing with their customers.
- Set up a reward and recognition system to reinforce the new paradigm
- Identify and eliminate non-value-added transactions.

Proponents of TQM and change via reengineering tend to point at experiences in other organizations which have been successful without providing an explanation of why the successes occurred. This is akin to the apocryphal story of the blind men examining an elephant by touch and describing the creature. Hawley states that the BPR and TQM initiatives are about organizational change (Hawley, 1995). While much is known about organizational change, there is no current theory of organizations which permits these approaches to be implemented in our current organizations in an effective manner. Our organizations are still generally based on adaptive modifications to the "Scientific Management" paradigm (Taylor, 1911). Think out of the organisation box and consider mapping the organisation onto the set of

processes it performs or should perform. A useful list of processes is provided in the International Standard ISO 15288 (Arnold, 2002).

4.1 Mapping the organization into processes and transactions

At the highest level, the organization can be represented as shown in Figure 4-1 which is identical to the drawing of the PPPT model shown in Figure 3-2. While the control element for the particular process is within the process element (Beer, 1972) drawing the boundaries for Figure 4-1 as discussed with reference to Figure 1-3 it can be seen that the organisation can be represented by the following two elements or two sub-systems of a single system:

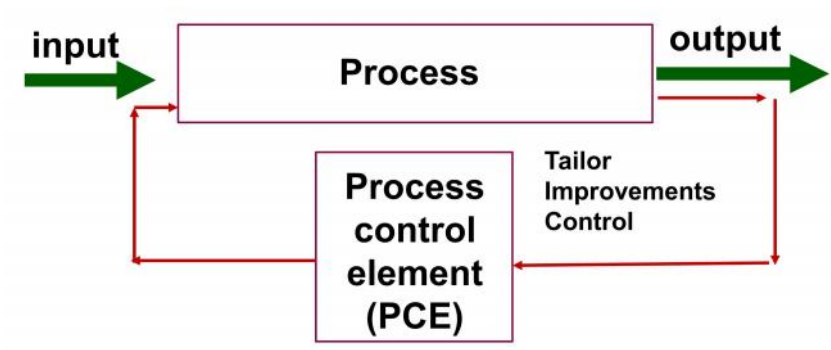


Figure 4-1 The organization as a process (ideal)

- **A process** which produces the products or services the organization sells to survive.
- **A process control element (PCE)** which supports the production process by planning, organising, directing and measuring the production process. This is the traditional management and support elements.

To analyse the organization from a transaction perspective, first decompose the single process block into several blocks in series representing stages in the production process as shown in Figure 4-2. Focus on what the process does, not how it is performed (Hammer and Champy, 1993) page 130). Focus on understanding it, not analysing it down to the smallest detail (Hammer and Champy, 1993) page 129). However, this representation lacks some kind of control or coordination between the process elements.

As an example of what can happen without the organizational control element, consider an organization producing statues of garden gnomes. The first production process element in Figure 4-2 is the moulding process. Here the plaster is poured into the moulds and the basic statues are produced.

This unit was having problems with their yield. The plaster was sticking to the moulds and the statues were being broken as they were being removed from the moulds. The department decided to improve the process, held several meetings and determined they needed to coat the mould to minimize the sticking. "It's just like coating a baking pan before placing the cake in the oven" said one of the workers. They spent several Saturdays on their own time experimenting with coatings and determined the optimal coating. They then instructed purchasing to procure the coating and implemented the change. The results exceeded their expectations, the yield increased to 99.8%, sticking was a thing of the past and breakages were reduced to 0.02% of the statues. However, at the time their yield went up, the next production process element (the paint shop) began to have problems and their yield went down. For some reason, the paint was smearing and taking a much longer time to dry. When the paint shop investigated the symptoms, they found the root cause was a coating of oil on the statues. In this situation, while the moulding department reduced their defects, they did it at the expense of the painting department.

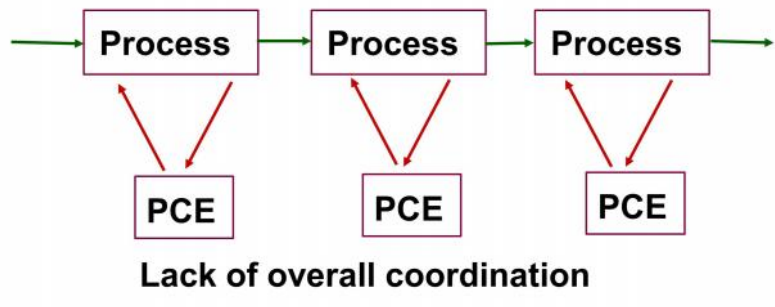


Figure 4-2 Multiple processes in series

There needs to be some coordination between the process elements to avoid this situation and optimise improvements over the entire process. If the organization is large, there may have to be several layers of this type of additional coordination as shown in Figure 4-3. When Figure 4-3 is turned upside down, as shown in Figure 4-4, the drawing appears as the traditional hierarchical organization.

4.2 Identifying customers

Each box in Figure 4-4 contains a process and a control element and each process element interfaces with other process elements by means of transactions. A product is transferred. Thus, each of the lines between the boxes in Figure 4-4 identifies a transaction of some kind. Any element sourcing a

transaction is a supplier; any element receiving a transaction is a customer. The location of suppliers and customers within and associated with your organization can be seen by mapping Figure 4-4 onto your particular organization chart and then analysing the transactions. There are two types of customers, namely:

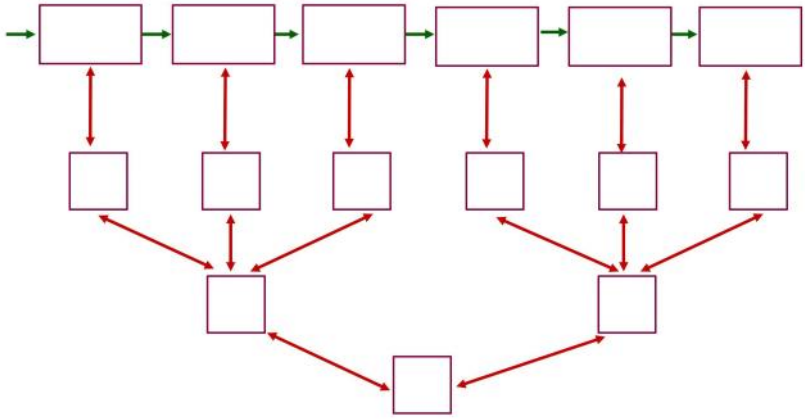


Figure 4-3 Ensuring control is effective

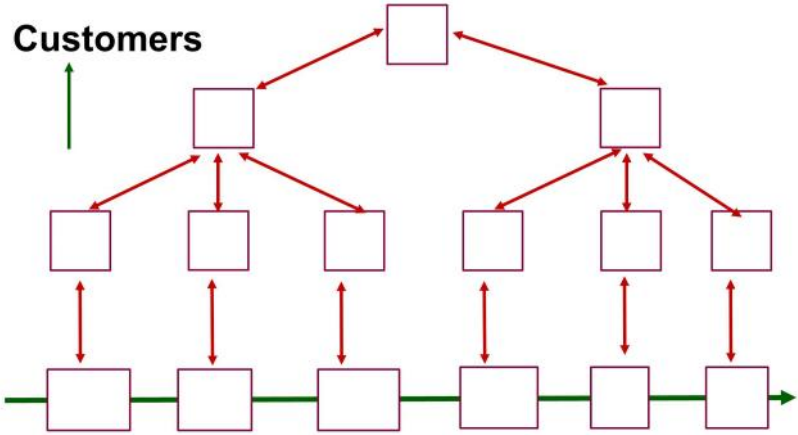


Figure 4-4 Traditional hierarchical organization

- **Vertical** - the traditional reporting path in which the customer is the supervisor (manager).
- **Horizontal** - the production process path in which the customer is the next link in the production process.

Each element in the organization may have either or both types of customers. For example, the elements in the actual production process have both vertical and horizontal customers, while the traditional middle management elements may only have vertical customers.

4.3 Identifying metrics

Little has been published on the topic of identifying the right metrics, because the right metrics make your organization more cost effective and consequently provide you with a competitive edge. "*The difference between winning companies and losers is that winning companies know how to do their work better*" (Hammer and Champy, 1993) page 26). The way to identify and use appropriate metrics is a multi-step process as follows:

- Determine what constitutes a metric.
- Determine what the effect of the measurement is to accomplish.
- Identify a proposed metric.
- Use the metric to make a measurement.
- Examine the effect of using the metric. If the change is positive (heading in the desired direction), then keep using it. If the change is negative (undesired), determine if the metric is at fault, or if the measurement approach is wrong and take appropriate corrective action.

4.4 Metrics

The ideal unit of measure (Juran, 1988) pages 76-78:

- **Provides an agreed basis for decision making** - Different people view things differently, and have different priorities. The metric must allow a meeting of minds.
- **Is understandable** - Metrics may not be understandable, perhaps because words do not have standardized meanings, or may require an educational background that is lacking.
- **Applies broadly** - For use to determine if an improvement has occurred.
- **Is susceptible to uniform interpretation** - The units used and types of errors must have been defined with appropriate precision.
- **Is economical to apply** - There is a trade-off between the cost of making the measurements and the value of having them. The cost may depend on the precision, so care must be taken to specify the correct precision.
- **Is compatible with existing designs of sensors** - If you can't measure it, there is little point in defining it as a metric.

4.5 Guidelines for identifying metrics

Guidelines for identifying metrics may be developed from the following

graphical representation of the "Cost of Quality" (Kasser, 1995). A process is performed by the implementation of a sequence of actions, by a number of people, using available resources over a specific period of time. A process can be shown in the form of a Gantt chart and a PERT chart as shown in Figure 4-5. There are three major milestones:

- The start point (S) where the process begins.
- The test point or check point (T) which determines the degree of conformance of the products which have been built, to their specifications.
- The end point (E) which occurs when the customer has accepted the products.

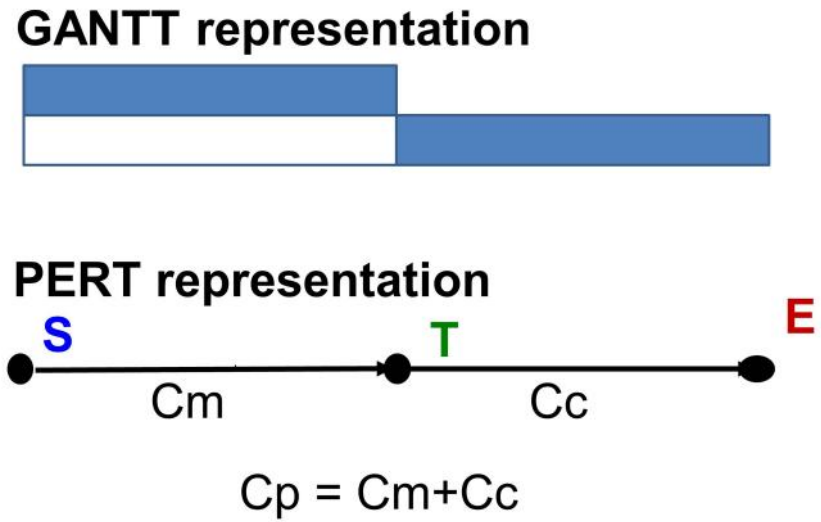


Figure 4-5 The generic process

Figure 4-5 may also be viewed as a vector representation of the costs of the process. The process contains two categories of costs, the costs of manufacturing (C_m) and the costs to complete after the checkpoint (C_c). Each category may contain fixed and variable costs depending on the specific situation. The total cost of the process (C_p) is the sum of the costs in each category, namely C_m+C_c.

Now consider this representation as the cost of the baseline zero-defect process. The first category of metrics is those than can be used to reduce the cost of the baseline process without reducing the quality of the product. The effect of using these metrics on the line shown in Figure 4-5 is to shorten the length between Points S and E.

The typical real world process however, produces products containing defects and can be depicted as shown in Figure 4-6 (Kasser, 1995) page 105). When the activity begins, it proceeds in a direction away from the baseline.

The checkpoint now lies at Point T1 which at the same distance from Point S as Point T, but in a different direction. The typical cost to complete is represented as a line between Points T1 and E. This cost contains two elements, namely:

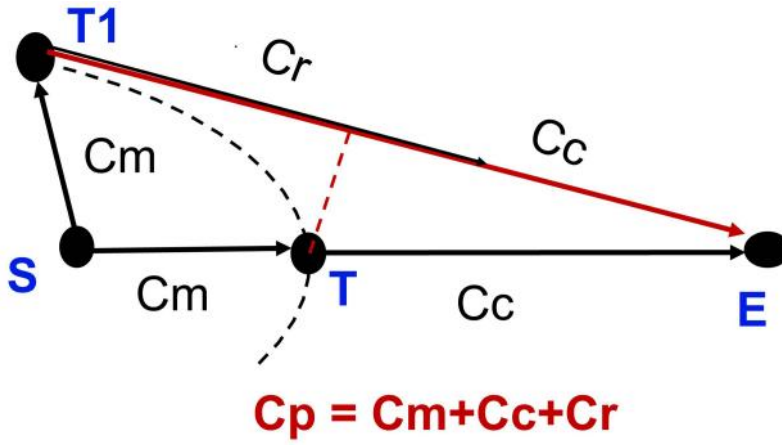


Figure 4-6 Cost of typical process with defects

- Cost to complete after the checkpoint (Cc). These are the same as those in Figure 4-5.
- Additional costs to complete due to not being on the baseline (Cr) which represents the cost of not doing it right in the first place, i.e., the cost of Quality.

The second category of metrics is those that can be used to reduce the number of defects. The effect of developing and applying these metrics is to push the line between Points S and T1 in Figure 4-6 down toward the baseline.

Both categories of metrics must be used when attempting to identify metrics for transactions. It is easier to identify items to measure in the horizontal dimension or production process because when there is a physical handover of objects, the attributes of objects can be measured. In the garden gnome example, measurements can be made of the:

- Number of statues passing through the production process.
- Defective statues and other wastage of raw materials.

In the vertical dimension, the transactions tend to be control and status information pertaining to lower levels in the organizational structure and directives being passed down from upper management. For status transactions, you can develop metrics based on the accuracy, relevance, correctness and timeliness of the information. For control transactions you can develop

metrics based on the clarity, conciseness, relevance and timeliness of the information.

4.6 Developing the metrics

The first set of transaction metrics to develop is the defect reduction metrics. These act within the current state of the organization and are based on an analysis of the individual horizontal and vertical transactions. In general these will be incremental improvements to the process and provide quick results.

Determining the correct metrics is critical. You tend to get what you measure, so the perspective is important. For example, years ago when smoking was permitted in public places two seminary students were studying together in the library when one said to the other "I'm dying for a smoke, do you think the Father will permit me to smoke?"

"I don't know" was the reply, "why don't you ask?"

So up got the student and went over to the Father. "Father is it permitted to smoke while studying," he asked.

"Certainly not!" said the Father.

Dejectedly the student returned to his seat and related what had happened.

"You asked the wrong question," said his friend, "let me have try".

So he got up, went over to the Father and asked "Father, is it permitted to study while smoking?"

"But of course" came the response.

Later, once the metrics are in place, you can develop the baseline cost reducing metrics based on an analysis of the value added by each process box in the transaction path when looking at the whole path as a system. This section of the change in general takes longer, and requires Reengineering and other major changes. It is where the re-organization begins and the resistance to change will be seen.

4.7 Identifying the non-value adding process element

Each process element in the organization should provide some economic added value along the horizontal or vertical path. This means that the cost of producing the added value (return on investment (ROI)) for the process element must be less than the added value itself. When the organization is mapped into a chart typified by Figure 4-4 and the inputs and outputs for each process element are measured, the change of value along the path may be seen. In the example, there is a change from raw materials to basic statues, to painted statues, then to statues ready for shipment, etc.in the horizontal path. However, the addition of value in the vertical path may not be

readily identifiable. In general, you are likely to observe pockets of questionable functions, duplicated functions and other tasks that have nothing to do with meeting the customer's needs (Hammer and Champy, 1993) page 4). You must take care to find out why the apparently non-value adding process elements are present before attempting to discard them as there may be a valid but not readily discernible reason. One reason might be that they are left over from some prior organisational arrangement and were not deleted when no longer required. The product from this analytical process is a set of facts that show exactly what each element is producing in both vertical and horizontal paths. The costs associated with each element may be obtained from your cost accounting system and the ROI of the element calculated. This scenario becomes the baseline for the transition to a customer driven transaction-based organization.

Having identified non-value adding or low ROI elements in the organization do not downsize them without further analysis. The people in those elements may have non-documented knowledge that is vital to the transformation or even to the survival of the organization. Given appropriate motivation, they may also swiftly become proponents of change.

4.8 Identifying the reengineering plan

Once the vertical and horizontal flows are identified a Reengineering or transition plan can be outlined. Sets of vertical and horizontal organizational elements may be combined into a process structure and the process then simplified. The optimal way to do this is to set up a vision of a transformed organization (to be) alongside the current organisation (as-is) and slowly migrate in a controlled manner as discussed below.

4.9 The change process

Change in the context of this Chapter is equated with process improvement which is generally depicted as being implemented in a PDCA manner and drawn as shown in Figure 3-1. The use of "cycle" and "circle" imply that the organization assumes the same state periodically which leads to *activity-based thinking*. It may be true that the PIT performs each action periodically. However, once an improvement is incorporated, the process is different. The texts on the subject generally do not mention the need for baselines and configuration control. Consequently, the results tend to be chaotic in a large organization with several simultaneous improvement initiatives in operation. Process improvement must take place in a controlled manner and the changes in the process implemented at specific milestones. There must not be any moving baselines. A better way to depict the process improvement process is by means of the process improvement spiral shown in Figure 4-7. The process improvement spiral is based on the approach that hardware

engineers use to build electronic circuits, namely build a little and test (fix any problems arising from this build), build a little more and test, and so on. It is thus an iterative loop with baselines consisting of four steps (Kasser, 1995):

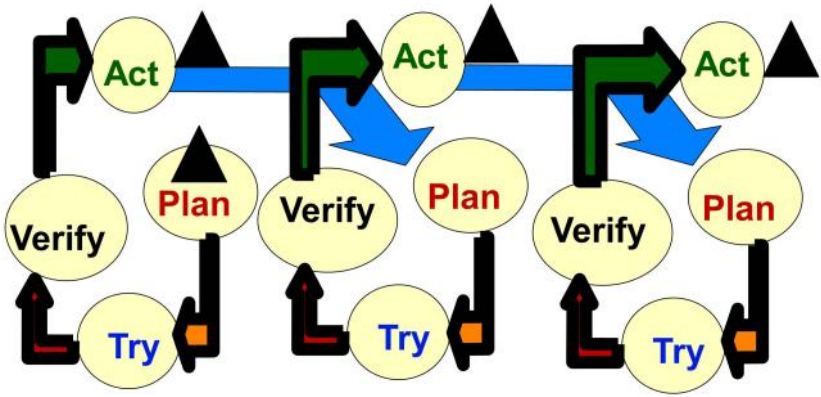


Figure 4-7 Process improvement spiral

- **Plan** - Define the process, analyse the process, investigate alternative actions, and propose improvements.
- **Try** - Try out the proposed improvement on a small scale.
- **Verify** - Measure the results, and verify that an improvement took place.
- **Act** - Assuming an improvement took place, upgrade the process to incorporate the improvement (new baseline), and iterate back to "plan" for the next go around.

The "iterative" term in the definition means that it is a continuous process in itself. The PDCA cycle goes back to at least 1913 when Henry Ford experimented on creating the flywheel magneto by what has now been called the assembly line. Ford wrote "*we try everything in a little way first- we rip out everything once we discover a better way, but we have to know absolutely that the new way is going to be better than the old before we do anything drastic*" (Ford and Crowther, 1922) page 81). This is the PDCA cycle in the form of change request, impact assessment (plan) try it out, check the change does what it promised to do, then act and implement the new process.

Process improvement has to be performed by a separate team of people working interdependently with the team who perform the process as stated in Section 3.3. The PIT interviews the process team (who have the best knowledge about the process but are usually under pressure to deliver the product), and gets full disclosure and suggestions for improvement from the process team using a soft systems intervention methodology such as CATWOE (Checkland, 1991). The PIT analyses the information and sugges-

tions from a system's perspective to determine the effect of the proposed improvement on all the sequential elements of the process. The process must then be upgraded at specific milestones in a controlled manner, rather than on an ad-hoc basis. Conceptually, upgrading a process is little different from upgrading a product release or making an organization change. Each action is an upgrade of a system and must be handled in an appropriate manner (process change requests, impact assessments and configuration control).

There are two types of improvements: - adaptive and innovative (Kirton, 1994). Adaptive improvements are more readily implemented than innovative ones since they improve the current paradigm. Innovative improvements tend to introduce a new paradigm, hence tend to be resisted. Adaptive improvements, however, also lead to the point of diminishing returns. This is the point where an innovative change is the only way to obtain any large degree of improvement as shown in Figure 4-8. These types of changes may also be directly related to cost. Adaptive improvements reduce costs over a time, yet the rate of reduction slowly reaches the point of diminishing returns. Note that while innovative changes may be employed anywhere along the curve, failure to innovate once the cost reduction curve flattens out tends to result in an organization going out of business.

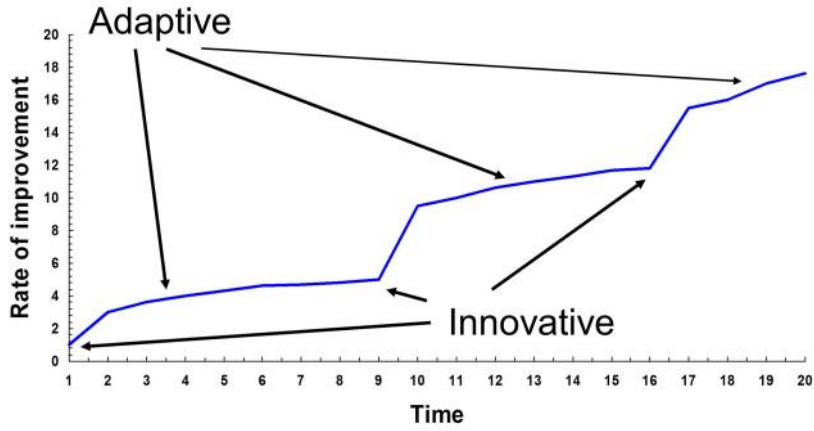


Figure 4-8 Adaptive and innovative changes

Consider the rate of improvement over time shown Figure 4-9. After seven time periods the rate of improvement has reached 8%. Forecast what rate can be expected in the next time period? Supposing the organisation leadership announces an improvement target of 20% for the following year. The initial reaction of the improvement team based on their forecast is that the announced target is impossible, and to send a memo back asking the leadership to stop setting arbitrary numerical targets. However, the target

may not be arbitrary. It may be that a competitor is about to announce a new product and if that numerical target is not achieved, the company will not be competitive in that product line with all the resultant consequences. Top management may be correct in that the new target is needed and setting a numerical goal. The PIT may be correct in that it is an impossible goal using their forecasts based on adaptive improvements, but they need to realise that it may be possible if an innovative change is made. When numerical targets are set that do not lie along the extrapolation of the historical curve of adaptive change, it is time to consider innovative change.

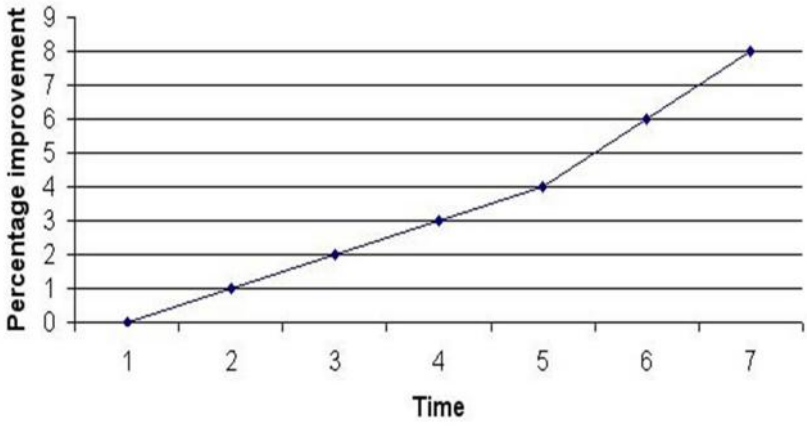


Figure 4-9 Improvements over time

Henry Ford wrote, "*Our policy is to reduce the price, extend the operations and improve the article. You will notice that the reduction of price comes first. We have never considered costs as fixed. Therefore we first reduce the price to a point where we believe more sales will result. Then we go ahead and try to make the price. We do not bother about the costs. The new price forces the costs down. The more usual way is to take the costs and then determine the price, and although that method may be scientific in the narrow sense, it is not scientific in the broad sense because what earthly use is it to know the cost if it tells you that you cannot manufacture at a price at which the article can be sold?*" (Ford and Crowther, 1922) page 146). It is a question of perspective and asking the right question. The usual question was "what does it cost to produce X?" from the alternative (out-of-the-box) perspective, the question was "how can X be produced for $Y?" This is but one example of the situation illustrated in Section 4.6 reflecting the situation in which we solve the problems we articulate. The key is to articulate the correct question, or in systems engineering terms defining the correct requirement.

While an organisation can reengineer at any time (Hammer and Champy, 1993), this same approach can be used to determine when an organisation has to reengineer to survive. Consider the reduction in cost over time shown in Figure 4-10. Costs have been decreasing but the curve is flattening out after seven months. Any further effort in process improvement will not achieve significant reductions. This is the point where an innovative change must be made, namely reengineer the process.

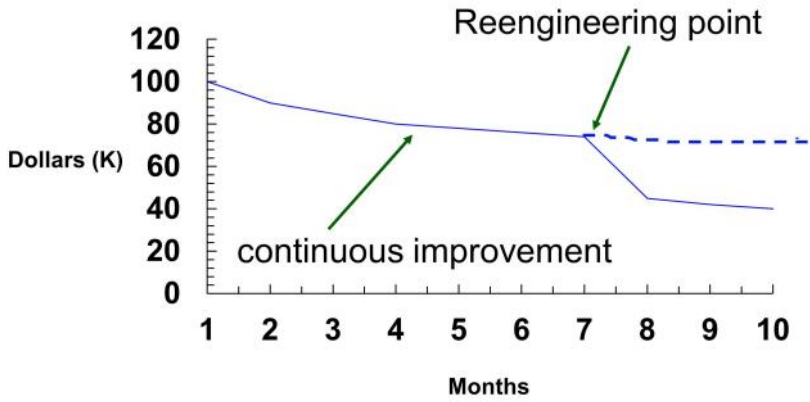


Figure 4-10 The reengineering point

There is no point in starting to change an organization unless you have a vision of what the changed organization will look like along the road to change as well as in its final form as shown in Figure 4-11. This picture does not need to be complete and 100% detailed, in fact it shouldn't be that detailed. This is because you will learn things during the process of change that will modify the vision because you will learn more about what is happening both within and without the organization. Don't forget, that while you are changing the organization, the outside world is also changing as shown in Figure 4-12. You can thus change the details of the vision at well-defined milestones along the road of change. In fact, you should plan to change the vision at these milestones. If you put together a plan for transitioning to meet today's needs, and it's going to take three years to make the change, at the end of the change, the organization will be three years out of date. The approach is to set up the change so that it gradually converges to the point where it's needed as shown in Figure 4-13 an approach also known as evolutionary acquisition.

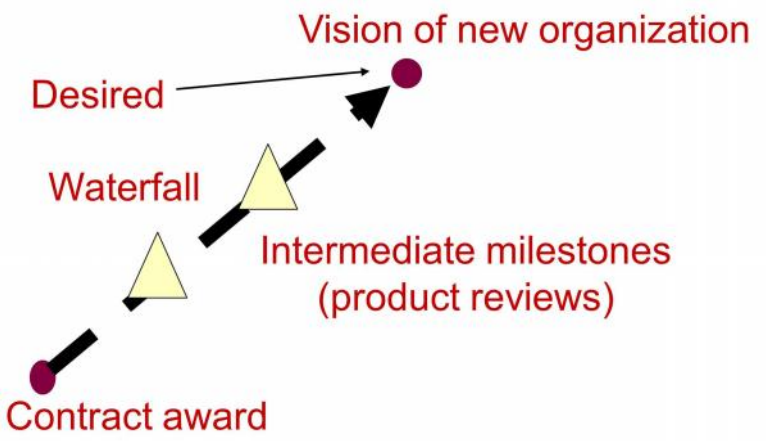


Figure 4-11 The Road to Change

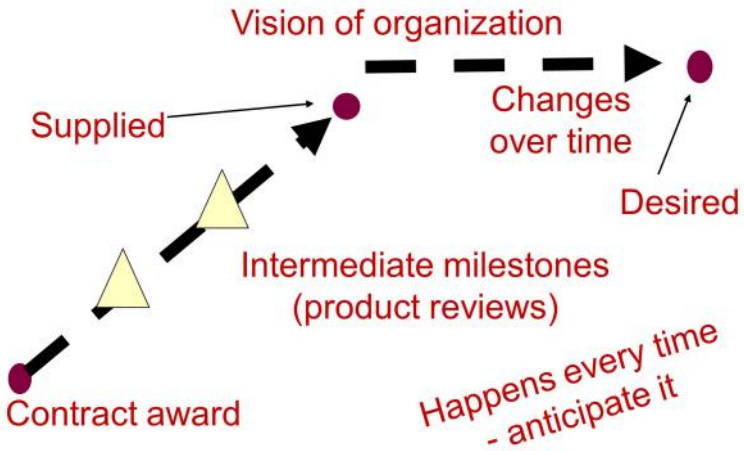


Figure 4-12 Chasing a moving target

The techniques used are based on the same budget tolerant approach to designing a major computerized system in the aerospace and defence industries (Denzler and Kasser, 1995)[17]. They are to:

- Prioritise the changes and then implement the changes in order based on the highest priorities.

[17] And discussed in Chapter 13

- Re-evaluate the priorities at the milestones and make the corresponding changes to the next set of activities.

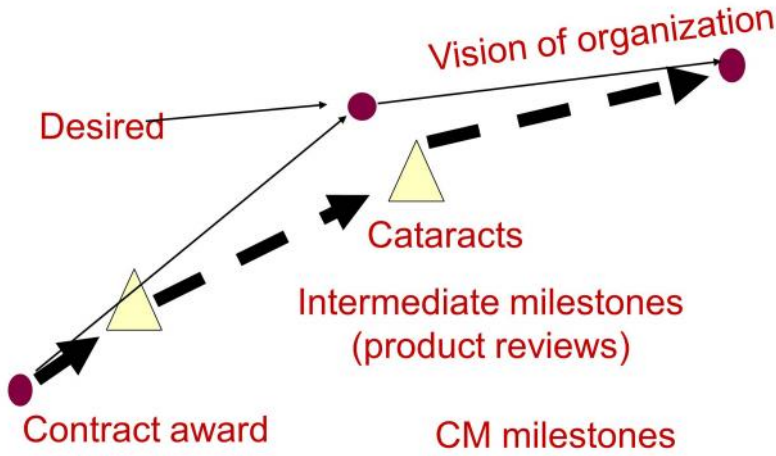


Figure 4-13 Convergence

While no two companies' business situations are identical (Hammer and Champy, 1993) page 159), there are methodologies that can be employed to perform the paradigm shift in an effective manner. **The change does not have to be chaotic.** Do not radically change everything in the organization at the same time. **The change must be gradual and made with care.** Start with a vision of the desired result and work backwards along the transition path via identifiable milestones to the present. For example, the sequence for implementing a transition from the current paradigm to the 'customer-driven process organization' paradigm is as follows:

1. Communicate the need to change.
2. Identify the current processes performed by the organization and determine the value chain.
3. Baseline the current state of the organization.
4. Create the draft vision statement of the new transaction-based organisation.
5. Create the transition plan.
6. Design the RRS.
7. Pilot one process transition to the new paradigm.
8. Implement the RRS.
9. Baseline the change.
10. Evaluate the experience
11. Update the transition plan
12. Start the transition cycle for the next process.

4.10 Support and resistance to change

Even where top management may be perceived as having the commitment to change, and can "communicate the vision," and the people at the lower levels in the knowledge organization are willing to try it, in many instances, middle management resist the change and the change fails. For example, in a survey of 1000 companies by Achieve International, more than 33% of the companies reported sabotage or internal resistance to these initiatives (Brecka, 1994). Most blamed middle managers for impeding quality (75%) and team efforts (70%). This resistance is because there seems to be nothing in it (the new system) for them. Thus Figure 4-14 charts resistance to change as a function of management level in the knowledge organization. In general:

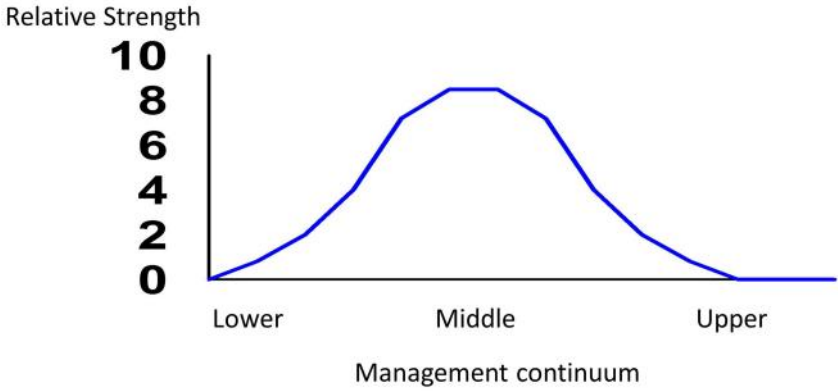


Figure 4-14 Resistance to change by management level

- Top management understand the need for change and desire the change.
- At the lowest level - the workers want the change and are frustrated by not being provided with the tools to implement the change.
- Middle management tends to resist the change and protect their own fiefdoms even though their actions are detrimental to the organisation as a whole.

You will encounter three types of people in the organization when implementing the change. They are those:

- **For the change** - who will make it happen if given the chance, whatever it takes. They tend to be the people involved with the process who can see the defects and want to initiate improvements. Implement the first change with these people. They will make it work. If you reward them visibly, you will set up the next batch of people to implement the next change.

- **Who are undecided** - they are sitting on the fence waiting to see which way the wind is blowing. The goal is to move them to your side of the fence so they support you.
- **Against the change** - they have no motivation to effect the change. The goal is to make them amenable to the change by first moving them to the undecided camp.

Think of the normal distribution curve in Figure 4-14 as showing a fence. Most people are sitting on the fence; your job is to encourage them to slide down the fence on your (the improved) side. If there is a real hard core of resistance, don't confront it early in the change process; bypass it for as long as you can, it may fade away on its own. The transaction analysis approach should tend to overcome the resistance at least for those personnel in non-value adding elements.

Telling people who had played a substantial role in creating an organisation that it is broken and needs major surgery is a sure way to invoke resistance to change. Approaches that ought to mitigate such resistance include:

- Employing an outsider who is not competing for promotion with employees as the process architect[18].
- Using variations of the phrase "the current organisation met the needs of the time; however a small modification is now necessary to meet the new situation" which does not assign blame it just notes that system that is no longer appropriate for the current situation.

4.11 Implement the reward and recognition system

The failure to institute an appropriate RRS is a major cause of most of the failures in BPR and TQM. The organization's whole RRS must reinforce the behaviour appropriate to the transition to the new organization (Harrington, 1995) page 469. This means that:

- The organisation needs a RRS as Drucker wrote "People in organisations, we have known for a century, tend to act in response for being recognised and rewarded-everything else is preaching. *The moment people in an organization are recognized-for instance by being asked to present to their peers what made them successful in obtaining the desired results-they will act to get the recognition. The moment they realize that the organization rewards for the right behaviour they will accept it*" (Drucker, 1993) page 195).

[18] See Chapter 19.

- The RRS evaluation criteria must reinforce behaviour that is in accordance with the values of the organization (Deming, 1993) namely, reflect the values of the *Excellence* organisation.

4.12 Summary

Making the transition to the *Excellence* (customer-based) organization is a difficult process generally characterized by resistance to change, chaos and inefficiency. However there is a better way and the transaction approach can be used to identify the non-value adding elements in the organization. Once they are identified, the process can be reengineered and implemented in a controlled manner. At this time, the process of reengineering into the new paradigm may receive lesser resistance than in conventional approaches, because it becomes in everybody's interest to be part of a value adding process element. The change must be reinforced with a change in the organization's RRS to reward behaviour appropriate to the new paradigm.

Your journey begins with the first few steps, and those steps are to identify and improve all the transactions within your organization.

1997

5 What do you mean you can't tell me how much of my project has been completed?

This Chapter looks at the organisation from the perspective of the SDLC for large systems which can take several years to complete. During this time, the:

- **Customer** makes periodic progress payments to the supplier. In this situation, since the acceptance tests are generally made at the end of the SDLC, the suitability of the product for its mission is unknown for the time in which the bulk of the payments are made.
- **Supplier** provides the customer with minimal information to demonstrate the risk of non-compliance with the Statement of Work (SOW). The information is typically provided in the form of:
    - **Management** - i.e., budget (estimated and actual), Gantt and PERT Charts, conformance to "best practices"
    - **Intermediate products** - i.e., documents, lines of code produced, defects found, number of requirements satisfied.
    - **Process** - i.e., degree of compliance to the appropriate Capability Maturity Model (CMM) and ISO models.

The intermediate reports are produced to reduce the risk of non-delivery and non-compliance to the requirements in the SOW. Now in spite of the measurements being made the supplier is unable to tell the buyer the exact percentage of completeness of the system under construction anytime during the SDLC.

5.1 Requirements

A definition of a successfully completed system is one that meets its requirements. However, measuring the percentage of completed requirements does not provide a measurement of completeness of the system for several reasons, including:

- **Nature of the requirements** - different requirements have different complexities, resulting in different implementation times and costs.
- **Changes in requirements over the SDLC** - the customer either does not state the full requirements for a system in the Request for Proposal (RFP), or changes them for various reasons during the SDLC.

The ideal SDLC is shown in Figure 5-1. A set of requirements for the system is developed based on the real need. The implementation phase of the SDLC is then supposed to take place across several milestones until the system is completed. Note that Figure 5-1 is identical to Figure 4-11, differing only by the words "organisation" and "system". This is because both figures represent changing needs over time.

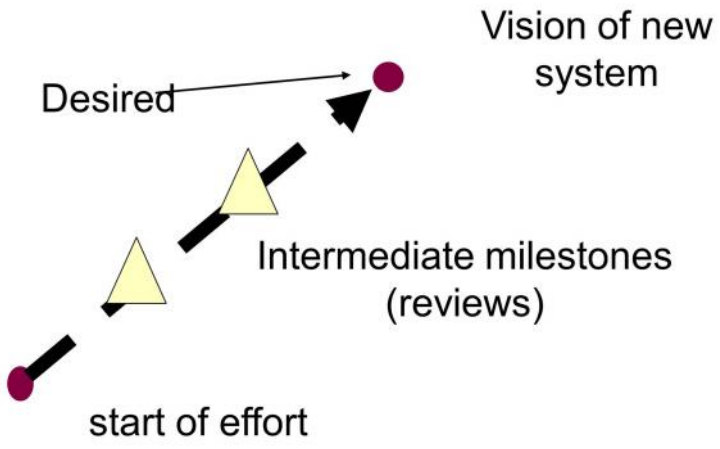


Figure 5-1 Ideal SDLC

The real world SDLC shown in Figure 5-2 is one in which the vision of the product (real need) changes during the implementation phase. Consequently, the requirements change. Thus, while the delivered system may meet its original requirements, if no change is made, the system will not meet the requirements in effect at the time of delivery. This situation leads to changes in requirements during the implementation phase, which in general are poorly controlled. And, the major consequences of failing to control changes are moving baselines and confusion leading to cost escalation and schedule delays (Kasser and Schermerhorn, 1994b). Again Figure 5-2 is identical to Figure 4-12. The situation is the same; the difference lies in the definition of the system being produced. In this instance it is a system being acquired, in Chapter 3 it was a new organisation being developed. Thus the process improvement methodology is the same as the methodology to develop any other product. The difference lies in the process that implements the methodology.

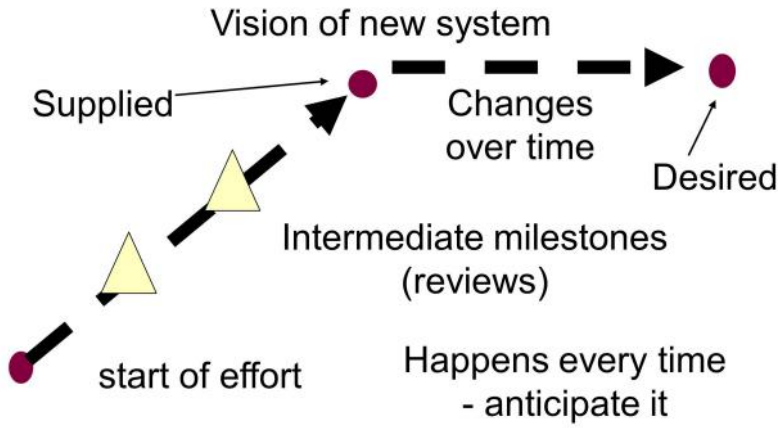


Figure 5-2 Actual SDLC

Recognizing that this situation was inevitable, the cataract approach (a series on mini waterfalls) shown in Figure 5-3 was proposed to control change (Kasser, 1995). The approach is best implemented using a budget-tolerant SDLC methodology based on the traditional waterfall SDLC model (Royce, 1970), but with significant enhancements[19].

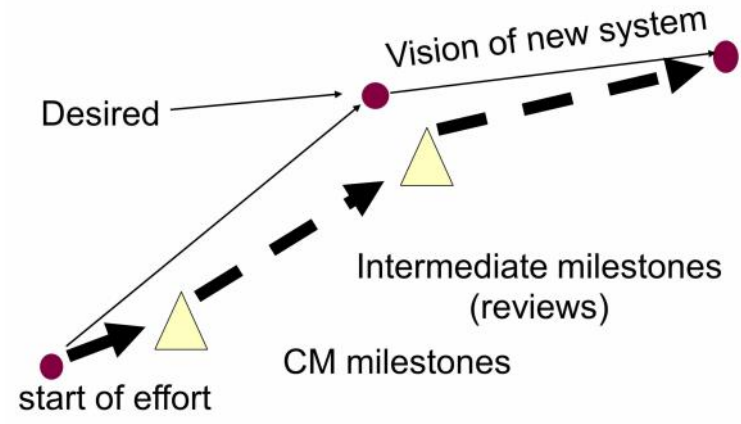


Figure 5-3 The anticipatory testing SDLC

Note the similarity of Figure 4-11, Figure 4-12 and Figure 4-13 to Figure 5-1, Figure 5-2 and Figure 5-3. This is because both products and organisations are systems whose requirements change while they are being developed, and the development process has to converge from meeting the re-

[19] See Chapter 13.

quirements at the time the process began to meeting the requirements at the time the process ends (delivery). Thus from this perspective, evolutionary acquisition is nothing new, it is the way we have always acquired systems. However, we have been using a flawed acquisition paradigm based on the assumptions that all the requirements must be known at the start of the acquisition process irrespective of how many years that process will take.

5.2 Categorized Requirements in Process

The budget tolerant methodology categorized requirements by cost (to implement) and priority. Tracking the implementation of the categorized requirements has led to a measurement approach that has the potential of providing a measurement of completeness of the product at any of the milestones in the SDLC. This measurement approach, called Categorized Requirements in Process (CRIP) (Kasser, 1999) is a way to estimate the percentage of project completion by looking at the change in the state of the requirements over time from several perspectives. CRIP charts use a technique similar to Feature Driven Development (FDD) (Palmer and Felsing, 2002) to monitor the state of a feature or requirement during the SDLC. FDD Charts however, show the state of every requirement or feature, namely they are suitable for detailed discussion by developers and testers, but provide information overload for managers and do not cover the entire SDLC. CRIP Charts on the other hand cover the entire SDLC and provide summaries suitable for management but have to be integrated into the process. The four-step CRIP approach is:

1. Categorize the requirements.
2. Quantify each category into ranges.
3. Place each requirement into a range.
4. Monitor the differences in the state of each of the requirements at the SDLC reporting milestones

The first part of the approach avoids the problem of comparing requirements of different complexities. The last step is the key element in the CRIP approach. Consider the four steps.

5.2.1 Categorize the requirements

Categories are identified for the requirements. Typical categories are:

- **Priority** of the requirement to the customer.
- **Complexity** of the requirement, i.e. the difficulty of implementing the requirement.
- **Estimated cost** to implement the requirement by the supplier.
- **Risk** - probability of occurrence, severity if it occurs, etc.

5.2.2 Quantify each category into ranges

Each category is then split into no more than ten ranges. Thus, for:

- **Priority** - requirements may be allocated priorities between one and ten.
- **Complexity** - requirements may be allocated estimated complexities between "A" and "J".
- **Estimated cost** to implement - requirements may be allocated estimated costs to implement values between "A" and "J".
- **Risk** – requirements may be awarded a value between one and five.

The ranges are relative, not absolute. Any of the several techniques for sorting numbers of requirements into relative ranges may be used. The buyer and supplier determine the range limits in each category. A requirement may be moved into a different range as more is learned about its effect on the development during the implementation phase. Thus, the priority of a specific requirement or the cost to implement may change between SDLC reporting milestones. However, the rules for setting the range limits must not change during the SDLC.

5.2.3 Place each requirement into a range

Each requirement is then placed into one range slot for each category. If all the requirements end up in the same range slot, such as all of them having the highest priority, the range limits should be re-examined to spread the requirements across the full set of range slots.

5.2.4 States of implementation

Each requirement can be in one of five states at any time during the SDLC. These states of implementation of each requirement during the project are:

- **Identified** - A requirement has been identified, documented and approved.
- **Working** - The supplier has begun work to implement the requirement.
- **Completed** - The supplier has completed work on the requirement.
- **In test** - The supplier has started to test the requirement.
- **Accepted** - The buyer has accepted delivery of part of the system (a Build) containing the implementation of the requirement.

The summaries of the number of requirements in each state are reported at project milestones.

5.2.5 Populating and using the CRIP Chart

Each cell contains three numbers, expected, actual and planned for next reporting period where:

- **Expected** - The number of requirements planned to be in the implementation state, based on the previous reporting milestone.
- **Actual** - The number of requirements in the implementation state.
- **Planned for next reporting period** - The number of requirements planned to be in the implementation state in the following reporting milestone.

For the first milestone reporting period, the values for "expected" are derived from the project plan for the time period. The "actual" value is the number measured during the reporting period, and the "planned for next reporting period" is a number derived from the project plan and the work done during the current reporting period. From then on the planned numbers are based on the state of the project. Numbers move horizontally across the CRIP Chart over time as shown in Figure 5-4. As work progresses the numbers flow across the columns from "identified" to "accepted".

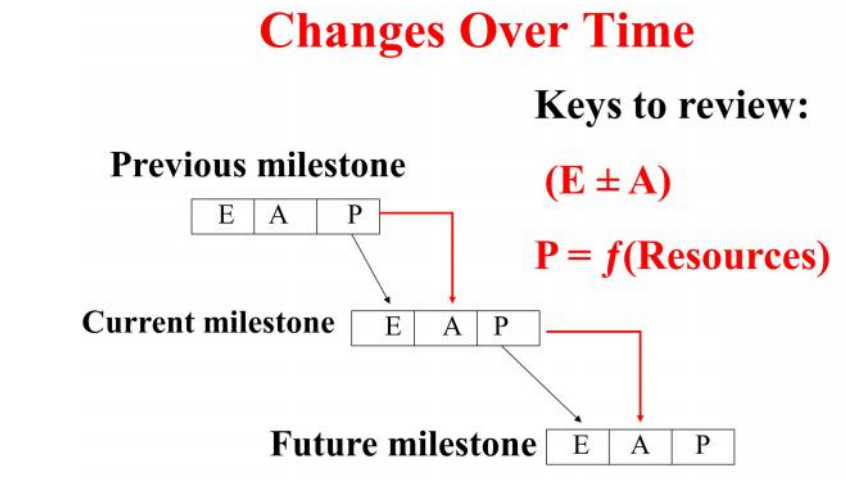


Figure 5-4 CRIP chart entry changes over time

At each reporting milestone, the *changes* in the *state* of each of the requirements *between* the SDLC reporting milestones are monitored. The states of each of the requirements in each of the categories are presented in tabular format (a CRIP Chart) at reporting milestones (major reviews or monthly progress meetings) as can be seen in the typical CRIP Chart shown in Figure 5-5. Colours can be used to draw attention to the state of a cell in the table. For example the colours can be allocated such that:

- **Violet** – shows requirements implementation is well ahead of estimate.
- **Blue** – shows requirements implementation is ahead of estimate.
- **Green** – shows requirements implementation is close to estimated values.

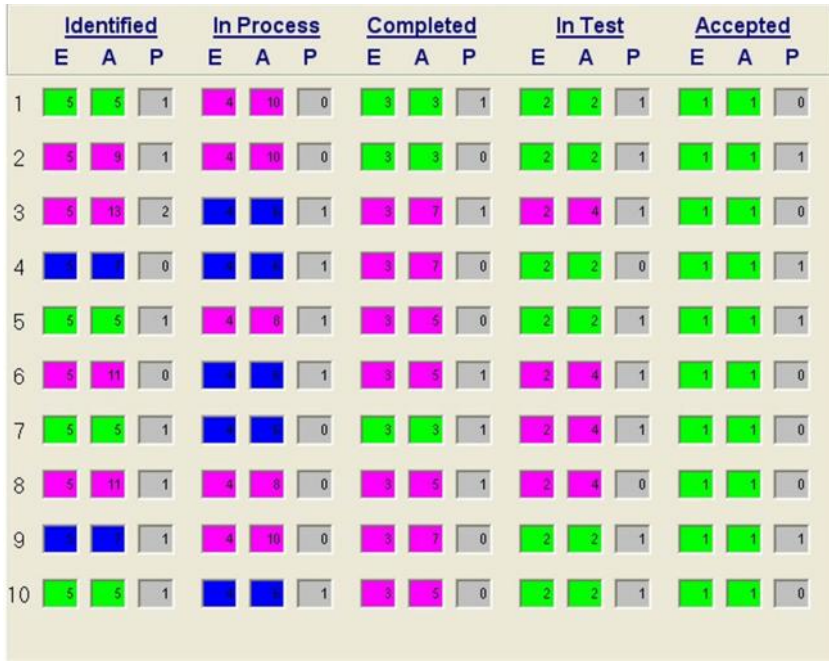


Figure 5-5 CRIP chart for category X

- **Yellow** – shows requirements implementation is slightly below estimate.
- **Red** – shows requirements implementation is well under the estimated value.

One chart can show that a problem might exist. Any time there is a deviation from expected to actual, the situation needs to be investigated. A comparison of the summaries from different reporting milestones can identify progress and show that problems may exist. On its own however, it cannot identify the actual problem. For example in Figure 5-5:

- The cell in the "Identified" column for Range 6 shows that the project planned that five requirements would be identified, but eleven requirements were actually identified and the project expects to identify none in the next reporting phase. Something may be wrong here!
- In Range 3, it was expected that the part of the system implementing two requirements would go into test in the last reporting period, yet four made it into test and one is planned for the next reporting period. Testing seems to be ahead of schedule.

The CRIP charts when viewed over several reporting periods can identify other types of "situations". The CRIP chart may be used on a standalone basis or in accordance with budget and schedule information. For example, if there is a change in the number of:

- **Identified requirements and there is no change in the budget or schedule**, there is going to be a problem. Thus, if the number of requirements goes up and the budget does not, the risk of failure increases because more work will have to be done without a change in the allocation of funds. If the number of requirements goes down, and the budget does not, there is a financial problem.
- **Requirements being worked on, and there is no change in the number being tested**, there is a potential supplier management or technical problem if this situation is at a major milestone review.
- **Requirements being tested, and there is no change in the number accepted**, there may be a problem with the supplier's process.
- **Identified requirements at each reporting milestone**, the project is suffering from requirements creep if the number is increasing. This situation may reflect controlled changes due to the change in the customer's need, or uncontrolled changes.

5.2.6 Advantages of the CRIP Approach

The advantages of the CRIP approach include:

- Links all work done on a project to the customer's requirements.
- May be used at any level of system decomposition.
- Provides a simple way to show progress or the lack of it, at any reporting milestone. Just compare the expected and actual numbers and ask for an explanation of the variances.
- Provides a window into the project for top management (buyer and supplier) to monitor progress.
- Identifies the probability of some management and technical problems as they occur, allowing proactive risk containment techniques.
- May be built into requirements management, and other computerized project and design management tools.
- May be built into Government contracts via the SOW. Falsifying entries in the CRIP Chart to show false progress then constitutes fraud.
- Requires a process. Some organisations don't have one, so they will have to develop one to use CRIP Charts.
- Requires configuration management which tends to be poorly implemented in many organisations. The use of CRIP Charts will force good configuration management.

5.2.7 Disadvantages of the CRIP Approach

The CRIP approach has the following disadvantages, it:

- Is a different way of viewing project progress.
- Requires categorization of the requirements.

5.2.8 Perspectives on CRIP Charts

Consider the CRIP charts from the following perspectives:

- Contractor past performance.
- Identifying the critical chain.

5.2.8.1 Contractor Past Performance

The CRIP chart numbers at major milestones can provide objective past-performance evaluation criteria and force cost effective behaviour. Consider the following examples.

- **Requirements are met or they are not.** Waived requirements are "not accepted requirements" by definition. Hence the process of "waiving requirements that a supplier cannot meet at the end of a project" shows up in the CRIP Chart when the planned number of accepted requirements at the pre-completion milestone is different from the number of requirements accepted at the completion milestone.
- **Requirements Creep.** Requirements creep shows up on the CRIP Chart in the number of identified requirements at major milestones. If requirements creep is a negative evaluation criterion in a short duration cost plus contract, then it is in the supplier's interest to identify a full set of requirements as early in the program as possible. The supplier is now motivated to get it right the first time.
- **Degree of completion.** It may be possible to develop a CRIP rating based on the difference between the number of system requirements identified over the SDLC, and the number accepted at the completion of the project and the total cost of the project as a function of the number of categories and ranges within each category. However, this rating will require a "CRIP standard" for future contracts.
- **Problems with the buyer's project team.** If the CRIP Charts show that all the buyer's requirements are met, yet the subjective past performance rating is poor, there may be a problem with the buyer's project team. This is something the customer should investigate.
- **Time will tell.** If requirements can only be tested over time, such as mission effectiveness requirements and failures, the buyer can update the final CRIP Chart in the past performance database (if they are included) to reflect the status of the requirements after several months of use.

5.2.8.2 Identifying the Critical Chain

If CRIP Charts are used to monitor the flow of work in a process, bottlenecks should show up where the process flow is constrained. These identify the critical chain according to the Theory of Constraints (Goldratt, 1990).

5.3 Conclusions

The CRIP Chart approach to measuring progress can provide a more accurate answer to the buyer's question than any other measurement approach in use today. It provides a high degree of visibility of the status of a project in both the buyer and supplier organizations that should discourage poor management in both organizations. However, it still does not guarantee the completeness of the system level requirements.

1998

6 What do you mean you can't tell me if my project is in trouble?

This Chapter looks at the context or background to the SDLC. Anecdotal evidence suggests that most projects do not fail due to the non-mitigation of technical risks. Rather, they fail as a result of poor management of the human element (Harrington, 1995; Deming, 1986). In addition, while the Standish Group identified ten major causes for project failure along with their solutions, they also stated that it was unclear if those solutions could be implemented (VOYAGES, 1996). This Chapter describes the development of a set of risk-indicators based on the human element. These risk-indicators can then be further refined into metrics to predict project failures.

The SDLC for large systems can take several years to complete. During this time, the:

- **Customer** makes periodic progress payments to the supplier. In this situation, since the acceptance tests are only made at the end of the SDLC, the suitability of the product for its mission is unknown for the time in which the bulk of the payments are made.
- **Supplier (contractor)** provides the customer with information to reassure the customer that the requirements in the SOW are being implemented. The information is provided in the form of:
    - **Management information** - i.e., budget (estimated and actual), Gantt and PERT Charts, conformance to "best practices."
    - **Intermediate products** - i.e., documents, lines of code produced, defects found, number of requirements satisfied.
    - **Process** - i.e., degree of compliance to the CMM and ISO models.

The reports containing this intermediate information are produced to demonstrate a low risk of non-delivery and non-compliance to the requirements. Drucker wrote that "*throughout management science, in the literature as well as in the work in progress, the emphasis is on techniques rather than on principles, on mechanics rather than on decisions, on tools rather than on results, and, above all, on the efficiency of the part rather than on*

the performance of the whole" (Drucker, 1973) page 509). Nothing seems to have changed in 30 years. While the SDLC has evolved from the waterfall method through various iterative approaches (e.g., Incremental, and Rapid Prototyping), the focus of measurements being made in today's paradigm are still in the process and product dimensions of the activities (Chapter 5) namely:

- **The process dimension** - Measurements are made of compliance to standards such as ISO 9001 and the CMM.
- **The product dimension** - Measurements (e.g., defects, lines of code per day, etc.) provide post-facto information, namely they report on what has happened and cause management to react to the reports.

These measurements provide post facto information, namely they report on what has already happened. This causes management to be reactive instead of being proactive. In addition, in spite of all the measurements being made, the supplier is often unable to tell the customer:

- The exact percentage of completeness of the system under construction anytime during the SDLC.
- The probability of successful completion within budget and according to schedule.

Thus, there is little wonder that software projects tend to fail (exceeds original estimates for cost and schedule, or terminate prematurely). However, the growing international dependency on the ISO standards for the SDLC indicates that this phenomenon of software project failure is not limited to the United States (US).

6.1 A methodology for developing metrics for predicting risks of project failures

The methodology for developing metrics for predicting risks of project failures has its origins in a class on Software IV&V in the Graduate School of Management and Technology at UMUC[20] in 1997 and 1998. Students in those classes wrote and presented term papers describing their experiences in projects that were in trouble. The term papers adhered to the following instructions:

1. Document a Case Study based on personal experience.
2. Analyse the scenario.

[20] These students were employed in the workforce and were working towards their degree in the evening. Their employment positions range from programmers to project managers. Some also had up to 20 years of experience in their respective fields.

3. Document the reasons the project succeeded or ran into trouble.
4. List and comment on the lessons learned from the analysis.
5. Identify a better way with 20/20 hindsight.
6. List a number of situational indicators that could be used to identify a project in trouble or a successful project while the project is in progress.

Once the papers had been graded, the methodology for developing metrics for predicting risks of project failures:

- Summarized the student papers to identify common elements called "risk-indicators".
- Surveyed systems and software development personnel via the Internet to determine if they agreed or disagreed with the risk-indicators.
- Summarized, analysed and validated the results.

6.2 Summary of student papers

Nineteen students produced papers that identified a total of 34 different risk-indicators. Each risk-indicator identified was a risk or a symptom of a risk that could lead to project failure. While several risk-indicators showed up in more than one student paper "poor requirements" showed up in all of the papers.

6.3 The survey

A survey questionnaire was constructed based on the student provided risk-indicators[21] and sent to systems and software development personnel via the Internet. Given there were 34 risk-indicators and expecting that recipients of the survey would not take the time to perform an Analytical Hierarchical Process (AHP) (Saaty, 1980) pair wise determination to identify the risk-indicators that were most and least important, an alternative prioritisation process was developed as described below. The survey questionnaire just asked respondents to:

1. State if they agreed or disagreed that the student provided indicators were causes of project failure[22].

[21] The students tended to state 'problems' using the semantics of 'solutions' or 'symptoms' rather than 'cause'.

[22] The authors of the survey recognized that there are other causes of (risks) project failure and added an "other" category to the survey questionnaire for "write-in" risks.

2. Pick out and prioritize the top seven risk indicators as causes of project failures.
3. List the seven risk-indicators that they thought were the least causes of project failures.

One hundred and forty-eight responses were received. The raw findings are summarized in Table 6-1. The first column contains a number identifying the risk-indicator described in the second column. The third column lists the number of students that identified the risk-indicator. The fourth column contains the percentage of agreement by the survey respondents. The fifth column contains the percentage of disagreement. The sixth column is the ranking of the risk-indicator. Thus for example, poor requirements ranked the highest as a cause of project failures with 97% agreement and 3% disagreement.

Table 6-1 Initial findings

Risk	Risk-Indicators	Students	Agree	Disagree	Rank
1	Poor requirements	19	97	3	1
2	Failure to use experienced people	7	79	21	13
3	Failure to use Independent Verification and Validation (IV&V)[23].	6	38	62	31
4	Lack of process and standards	5	84	16	11
5	Lack of, or, poor plans	4	95	5	2
6	Failure to validate original specification and requirements	3	91	9	3
7	Lack of Configuration Management	3	66	34	19
8	Low morale	2	51	49	24
9	Management does not understand SDLC	2	59	41	22
10	Management that does not understand technical issues	2	56	44	23
11	No single person accountable/responsible for project	2	69	31	18

[23] The papers were written by a class on IV&V, hence the emphasis on IV&V. However, if the descriptions of tasks that IV&V should have performed (in the papers) are examined, the word "IV&V" could easily be replaced with the word "systems engineering" and the papers would be equally valid.

Risk	Risk-Indicators	Students	Agree	Disagree	Rank
12	Client and development staff fail to attend scheduled meetings	1	42	58	28
13	Coding from high level requirements without design	1	75	25	14
14	Documentation is not produced	1	63	38	21
15	Failure to collect performance & process metrics and report them to management	1	48	52	25
16	Failure to communicate with the customer	1	88	12	5
17	Failure to consider existing relationships when replacing systems	1	85	15	10
18	Failure to reuse code	1	27	73	34
19	Failure to stress test the software	1	75	25	15
20	Failure to use problem language	1	34	66	30
21	High staff turnover	1	71	29	16
22	Key activities are discontinued	1	74	26	17
23	Lack of Requirements Traceability Matrix	1	67	33	19
24	Lack of clearly defined organizational (responsibility and accountability) structure	1	82	18	11
25	Lack of management support	1	87	13	6
26	Lack of priorities	1	85	15	8
27	Lack of understanding that demo software is only good for demos	1	47	53	26
28	Management expects a CASE Tool to be a silver bullet	1	45	55	27
29	Political considerations outweigh technical factors	1	86	14	9
30	Resources are not allocated well	1	92	8	4
31	The Quality Assurance Team is not responsible for the quality of the software	1	40	60	29
32	There are too many people working on the project	1	36	64	32

Risk	Risk-Indicators	Students	Agree	Disagree	Rank
33	Unrealistic deadlines hence schedule slips	1	86	14	7
34	Hostility between developer and IV&V	1	33	67	33

6.4 Survey results

The survey results were surprising. Modern TQM theory holds that the Quality Assurance Department is not responsible for the quality of the software. Everybody shares that responsibility. Thus, while it was expected that most respondents would disagree with this risk-indicator, only 60% of the respondents disagreed. It was also anticipated that most respondents would agree with the other risk-indicators, yet the overall degree of agreement was:

0.7% (one respondent) agreed with all 34 risk-indicators.
8.1% agreed with at least 30 risk-indicators.
51% agreed with at least 20 risk-indicators.
93% agreed with at least 10 risk-indicators.
As for the degree of disagreement:
0.7% (one respondent) disagreed with 25 risk-indicators.
4.7% disagreed with at least 20 risk-indicators.
52% disagreed with at least 10 risk-indicators.
88% disagreed with at least one risk-indicator.

6.5 Further analysis

The basic assumption is that while there were 34 risk-indicators, the major effect on a project will be due to a few of them, so we are only concerned with identifying the most and least important Given there were 34 risk-indicators and expecting that recipients of the survey would not take the time to perform an Analytical Hierarchical pair wise determination to prioritise the risk-indicators that were most and least important, the top seven (high priority) risk-indicators were identified using the following approaches:

- The Talley
- Priorities
- Top Seven

6.5.1 The tally

The tally was performed as follows. For each survey response to a risk-indicator, an “agree” was allocated a value of +1 and a “disagree” a value of -1. The answers to each risk-indicator across the survey statements were

then tallied. The risk-indicators receiving the highest positive values (most agreement) as causes of project failure are shown in Table 6-2. Poor requirements came out as the top reason.

Risk	Risk-indicator	Tallied Responses
1	Poor requirements	134
5	Lack of, or, poor plans	125
6	Failure to validate original specification and requirements	113
30	Resources are not allocated well	109
16	Failure to communicate with the customer	106
25	Lack of management support	98
33	Unrealistic deadlines hence schedule slips	97

Table 6-2 The talley results

6.5.2 Priorities

The survey questionnaire asked respondents to pick out and prioritise what they thought were the top seven risk-indicators. Since a high priority was represented by a low number (Priority of 1 was the highest), for each response, the risk-indicator was allocated (7- the priority) and the total allocation for that risk-indicator across all the surveys summed as a weighted response. The weighted results are for the top seven risk indicators are shown in Table 6-3. Again poor requirements came out as the biggest contributor to project failures.

Risk	Risk-indicator	Weighted response
1	Poor requirements	864
16	Failure to communicate with the customer	683
5	Lack of, or, poor plans	574
4	Lack of process and standards	361
25	Lack of management support	350
6	Failure to validate original specification and requirements	329
29	Political considerations outweigh technical factors	304

Table 6-3 The results by priority (top priority first)

6.5.3 Top seven

Since the actual position may be subjective, the number of times a risk-indicator showed up in the priority list was also computed. The top seven were extracted and are shown in Table 6-4. These results show a high degree of consensus on these risk-indicators as causes of project failures. There is a difference in the order of the sixth and seventh risk-indicator, but the numbers are too close to draw any other conclusions.

Risk	Risk-indicator	Count
1	Poor requirements	99
16	Failure to communicate with the customer	86
5	Lack of, or, poor plans	77
4	Lack of process and standards	51
25	Lack of management support	51
29	Political considerations outweigh technical factors	45
6	Failure to validate original specification and requirements	44

Table 6-4 Top seven causes

Risk	Risk-indicator
5	Lack of, or, poor plans
8	Low morale
15	Failure to collect performance & process metrics and report them to management
25	Lack of management support
27	Lack of understanding that demo software is only good for demos
29	Political considerations outweigh technical factors
32	There are too many people working on the project
33	Unrealistic deadlines hence schedule slips

Table 6-5 Risk indicators with little differences between perception of managers and non-managers

6.5.4 Sensitivity to management experience analysis

The sample size for respondents without management experience was 99. The Tally responses for the risk-indicators were examined to see if there was a difference in the responses between non-managers and managers with various years of experience. Differences of less than 10% were noted for the risk-indicators listed in Table 6-5. Since "poor requirements" don't show up in this list, there was a difference of opinion as to whether poor requirements constituted a risk-indicator in the responses between non-managers and managers with various years of experience.

6.5.5 The "other" category

Several respondents added a small number of risk-indicators in the "other" category of the questionnaire. These additional risk-indicators were

- Failure to control change.
- Rapid rate of change of technology.
- Low bidding to buy into contracts.
- Poor management.
- Lack of a technical leader.

Thus, the small student sample size of 19 seems to have identified most of the important risk-indicators.

6.5.6 The risk-indicators most people disagreed with

Part of the analysis of the survey results was to determine which risk-indicators received the most amounts of disagreement as causes of project failure. This was done in two ways by determining the:

- Largest number of disagreements by the recipients.
- Least number of agreements by the recipients.

The risk-indicators receiving the largest number of disagreements are shown in Table 6-6 and the risk-indicators that received the least number of agreements as causes of project failure are shown in Table 6-7. In each method of analysis, six risk-indicators showed up in the group receiving the most amount of disagreement, namely:

- **Failure to reuse code:** A major advantage of object-oriented technology is the ability to lower costs by reusing code. Yet 73% of those surveyed did not agree with this risk-indicator.
- **Hostility between developer and IV&V:** This risk-indicator shows a team problem and results in less than optimal costs due to the lack of cooperation.

Risk	Risk-indicator	Responses
18	Failure to reuse code	88
3	Failure to use Independent Verification and Validation (IV&V)	80
32	There are too many people working on the project	75
12	Client and development staff fail to attend scheduled meetings	74
34	Hostility between developer and IV&V	70
31	The Quality Assurance Team is not responsible for the quality of the software	68
15	Failure to collect performance & process metrics and report them to management	67

Table 6-6 Risk-indicators receiving the largest number of disagreements

Risk	Risk-indicator	Responses
20	Failure to use problem language	30
18	Failure to reuse code	32
34	Hostility between developer and IV&V	34
32	There are too many people working on the project	43
31	The Quality Assurance Team is not responsible for the quality of the software	45
3	Failure to use Independent Verification and Validation (IV&V)	49
28	Management expects a CASE Tool to be a silver bullet	53
12	Client and development staff fail to attend scheduled meetings	4
27	Lack of understanding that demo software is only good for demos	55

Table 6-7 Risk-indicators receiving the least number of agreements as causes of project failure

- **There are too many people working on the project:** This risk-indicator supports Fred Brooks who described the problems associated with assigning additional people to projects (Brooks, 1982).
- **Failure to use problem language:** The use of problem language was promoted as one of the major advantages by Ward and Mellor (Ward and Mellor, 1985). Yet, only 34% of the respondents agreed that failure to use it was a risk. Several did not know what the term meant.
- **The Quality Assurance Team is not responsible for the quality of the software:** As discussed above, this was the only indicator that should have shown disagreement.
- **Client and development staff fail to attend scheduled meetings:** This is a symptom of poor communication between the client and the developer. In addition, while there are other communication techniques available, if meetings are scheduled, and not attended, negative messages are sent to the project personnel.
- **Failure to collect performance & process metrics and report them to management:** If measurements are not made and acted upon, how can the process be improved? Yet 52% of the respondents disagreed that this was a risk-indicator.

Risk	This study	CHAOS study
1	Poor requirements	Incomplete requirements
16	Failure to communicate with the customer	Lack of user involvement
30	Resources are not allocated well	Lack of resources
	-	Unrealistic expectations
25	Lack of management support	Lack of executive management support
	-	Changing requirements and specifications
5	Lack of, or, poor plans	Lack of planning

Table 6-8 The correlation between this study and the CHAOS study

6.6 The CHAOS study

The Chaos study mentioned in Chapter 1 served as a reference to validate this study (CHAOS, 1995). The Chaos study had identified some major rea-

sons for project failure. The five risk-indicators in this study that were chosen as the most important causes for project failure also appear on the CHAOS list of major reasons for project failure. The correlation between this study and the CHAOS study is shown in Table 6-8. While "resources are not allocated well" did not show up in the top seven lists of this study, it was fourth in the Tally. Thus, this study supports the findings of the CHAOS study.

6.7 Presence of risk-indicators in ISO 9001 and the software-CMM

The elements of Section 4 of the ISO 9001 Standard and the five levels of the Software-CMM (CMM, 1995) were examined and interpreted to determine if the top student identified risk-indicators were covered in the ISO Standard and in the Software-CMM. The ISO 9001 Standard defines the minimum requirements for a Quality system, while the Software-CMM tends to address the issues of continuous process improvement more explicitly than does the ISO 9001 Standard. The findings are shown in Table 6-9 where an 'X' represents the presence of the indicator. The same two major risk-indicators could not be mapped into either the elements of Section 4 of the ISO Standard, or the Software-CMM, namely:

- [Risk 29] Political considerations outweigh technical factors.
- [Risk 33] Unrealistic deadlines hence schedule slips.

Thus, conformance to either or both Quality standards does not ensure mitigating these risks.

6.8 The development of metrics to identify the presence of these indicators

The large consensus on the major reasons for project failure seems to show that if we could remove these reasons, projects would have a greater probability of success. The Chaos study showed that projects tended to succeed if the opposite of these risk-indicators were present (e.g., good requirements instead of poor requirements). The follow-up Voyages paper stated that the causes of project failures were known, but it was unclear if the solutions could be implemented The (VOYAGES, 1996). Thus it seems that the current metrics paradigm is focused on measuring the wrong things and needs to be changed to develop metrics to show the presence or absence of the major risk-indicators identified above and any other known major causes of project failures. So consider ways metrics can be developed for the following risk-indicators:

Table 6-9 Comparison with ISO 9001 and CMM

Risk	Section 4.x of ISO 9001																				Software-CMM				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	1	2	3	4	5
1	o	o	x	x	x	x	o	x	o	o	o	o	x	o	o	o	o	o	o	o	o	x	x	o	o
4	o	x	o	o	x	o	o	x	x	x	o	o	o	o	o	x	x	o	x	x	o	x	x	x	o
5	o	x	o	o	x	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	x	x	o	o
6	o	o	x	x	o	x	o	x	o	x	x	o	o	o	x	o	x	o	x	o	o	x	x	o	o
16	o	o	x	o	x	o	o	o	o	o	o	o	o	x	o	x	o	o	o	o	o	x	x	o	o
25	x	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	x	o	o
29	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o
30	x	o	o	o	o	o	o	o	x	o	x	o	o	x	o	o	x	x	x	o	o	o	o	x	o
33	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o

- **Poor requirements:** Current requirements management tools do not indicate if requirements are defective. That is, if they violate the requirements for writing requirements (Kasser, 1995) page 166). This risk can be mitigated by teaching personnel how to write good requirements using techniques such as a Requirements Workshop (Kasser, 1995) page 259)[24]. Metrics can then be developed to measure the number of defects in requirements documents and provide a Figure of Merit for the document[25].
- **Failure to communicate with the customer:** This activity mitigates the risk of creating poor requirements and failing to cope with changes. This activity should be part of the process, is covered in ISO 9001 and should be measured. However, would just counting the number of meetings and telephone calls during the SDLC provide useful information?
- **Lack of, or, poor plans:** A project plan was recognized as a critical document as far back as the US Military Standard for Systems Engineering Management (MIL-STD-499A, 1974), yet many software projects still either do not have one, or ignore it. Since it must be a living document (Kasser, 1995) page 163), metrics need to be developed to monitor the use/updates of the plan during the SDLC.
- **Lack of process and standards:** These are covered by the Software-CMM and the ISO Standards covering the SDLC.
- **Lack of management support:** We can infer that projects need management support to succeed. We need to develop metrics to measure the degree of visible management support. This risk-indicator supports the prime reason that TQM fails if the application of TQM can be considered as a project.
- **Failure to validate original specification and requirements:** not performing this function means that the system being built may not meet the needs of the customer. Having a process and including this activity as a step in the process can mitigate this risk.
- Political considerations outweigh technical factors: Metrics need to be developed for this risk.
- **Resources are not allocated well:** This risk follows on from the lack of, and use of a plan. Having and using a plan will mitigate this risk.
- **Changing requirements and specifications:** Section 5.2 presented a CRIP Chart technique for monitoring progress and making both progress and

[24] Or software developed for the purpose (Kasser, 2004), (Kasser, Tran and Matisons, 2003), (Kasser, Scott, Tran and Nesterov, 2006) such as Tiger Pro shown in Figure 17-3.
[25] See section 17.6.2.

changes in requirements readily visible to top management and any other interested parties. The CRIP approach can easily be built into requirements management tools. However, just measuring the fact that the requirements are changing will not mitigate this risk unless a change tolerant development methodology is employed such as the Cataract Methodology (Chapter 13) using the task management approach (Chapter 15).

However, just changing the metrics paradigm may not be the complete solution. Cobb's Paradox states "*We know why projects fail, we know how to prevent their failure, so why do they still fail?*" (VOYAGES, 1996). Now a paradox is a symptom of a flaw in the underlying paradigm. Perhaps Deming and Juran provided the remedy. Deming stated that 94% of the problems belong to the system (i.e., were the responsibility of management) (Deming, 1993) page 35), and as already mentioned in Chapter 2, Juran was quoted by Harington stated that management causes 80 to 85% of all organizational problems (Harrington, 1995) page 198). In this survey, both managers and non-managers tended to disagree with the two management risk-indicators. Thus the there was a consensus that:

- [Risk 9] Management does not understand the SDLC, and
- [Risk 10] Management that does not understand technical issues were NOT causes of, or contributors to, project failures.

The survey respondents disagreed with Juran and Deming. It is difficult to understand how IT managers can make informed decisions to mitigate technical risks if they don't understand the implications of their decisions. The resolution of Cobb's paradox may be to use systems engineering to reengineer F. W. Taylor's work and develop a new paradigm for an information age organization which performs most of the functions of middle management without managers such as the one described in Chapter 3.

6.9 Deficiencies in the study

The following deficiencies are present in the study:

- The sample size is small.
- The level of expertise of the respondents is unknown.

6.10 Conclusions and recommendations

Except for *poor requirements*, none of the risk-indicators identified by this study are technical. Thus, the findings support:

- Resources spent mitigating technical risks are wasted unless the major risk-indicators discussed in this Chapter are also mitigated. Thus, it is critical to develop and use good metrics for them.

- The need for continual training to provide managers with the skills to become capable of effective technical management.

6.11 Areas for further study

This study raises some interesting areas for further study including:

- Determining which of the risk-indicators have a greater effect on the schedule and budget?
- Correlating the general agreement with the risk-indicators to specific ones. For example, did those who agreed with "political considerations outweigh technical factors" agree with more risk-indicators than those who didn't agree with it?

2000

7 The certified systems engineer – it's about time!

This Chapter looks at improving systems engineering from the perspective of the people who perform it, namely the systems engineers. Recognizing that a process is only as good as the people who perform it, this Chapter defines the requirement for the certification of systems engineers to establish a minimum level of competency for systems engineers. After a brief survey of how other disciplines certify their practitioners[26], this Chapter discusses the derived requirements for implementing the certification process and recommends an approach to prototype and administer the certification process for systems engineers.

7.1 Background

Recognizing that major government funded systems development has traditionally been characterized by cost and schedule overruns and other (spectacular) failures, attempts have been made to alleviate the situation.

Major systems development takes place over a period of years in a scenario wherein the customer makes periodic payments to the supplier based on the supplier's promise that all is well and the product is being manufactured on schedule and within budget. It is only after months or years of making these progress payments that the customer finds out that problems exist and large amounts of money may have been wasted.

In an attempt to minimize the risk of cost and schedule overruns, the customer has tried a number of different approaches to improving the product and the process.

- **The product** - Quality Assurance, Test and Evaluation, and Independent Verification and Validation (IV&V), try to ensure that the product not

[26] The process starts with an out-of-the-box approach.

only works, but is also the right product. Testing the intermediate products as they are being built does this. Deming was very much in favour of building quality into the product in the first place (Deming, 1986) page 11). He wrote, *"Defects are not free. Somebody makes them, and gets paid for making them"*.

- **The process** - "*Quality comes not from inspection, but from improvement of the production process*" (Deming, 1986) page 29). The customer has attempted to improve the supplier's processes by emphasizing "standards" and "best practices". The basic assumptions being that well-established processes, compliant to standards, minimize waste, and hence saves money. The most widely known examples of these standards are the International Organization of Standards (ISO) 9000 series and the system and software engineering CMMs.

However, a major cause of cost and schedule overruns has so far not been addressed, namely the people involved in the systems engineering process even though the situation has been recognized for a long time. For example, twenty years ago, Robert Frosch the then Assistant Secretary to the United States Navy, wrote, *"Systems, even very large systems, are not developed by the tools of Systems Engineering, but only by the engineers using the tools"* (Frosch, 1969).

Deming also wrote, *"People are part of the system"* (Deming, 1986) page 366).

This Chapter addresses the issue of improving the quality of systems engineering by improving its practitioners. It does this by applying models gleaned from other disciplines[27]. Systems engineering techniques were used in this study as described herein.

The problem of poor systems development was stated at the beginning of this Chapter. The next milestone was the realization that the people dimension was being ignored, yet in other disciplines there exist ways of guaranteeing the quality of the personnel. Consider these examples.

- **Management** - The Institute of Certified Professional Managers (ICPM) provides certification for professional managers.
- **Software Engineering** – The American Society for Quality (ASQ) provides various certifications for the quality disciplines. The state of Texas also recently certified software engineers.
- **Engineers** – The Institution of Electrical Engineers (IEE) in the United Kingdom offers the Chartered Engineer (CEng) designation for professionalism in engineers.

[27] Out-of-the-box thinking.

It is time for systems engineers to be certified in a similar manner to attempt to guarantee that the personnel involved in the systems engineering process are possessed of a degree of competency. The customers can then feel that the more certified systems engineers working on their major system development, the greater the probability that their funds will be expended in a more effective manner.

After much research in the non-systems engineering literature, it was found (Lawler III, 1973) -

Ability = f (Aptitude * (Training + Experience))

Where:

- **Ability** refers to how well a person can perform at the present time. If a person lacks the ability to accomplish a task, no amount of motivation or effort will lead to better performance (Kast and Rosenzweig, 1979).
- **Aptitude** refers to whether an individual can be brought through training and experience to a specified level of ability.

The purpose of certifying people as systems engineers is to ensure they are possessed of a minimum level of training expertise and experience. The employer can then address the issue of ability by providing the appropriate motivation. However, the certifying agency *cannot really certify the competence* of the person. The agency can however certify that *the applicant has shown proof of compliance with the requirements for certification.*

7.2 The certified systems engineer

Given the requirement for a Certified Systems Engineer, the problem then becomes to define the model used to meet the requirement for certifying systems engineers.

After further research analysing the similar situation in other disciplines which certify practitioners, the model proposed for certifying systems engineers is based on the Certified Manager (CM) designation by the ICPM in the field of management.

Having decided on the model, the problem then decomposes into the following derived problems.

1. What are the educational requirements?
2. What are the experience requirements?
3. How does the applicant for certification demonstrate compliance with the educational and experience requirements?
4. Who will administer the certification program?

Consider each of them in turn.

7.2.1 Educational requirements

The education requirements must be set to for the applicant to demonstrate an understanding of what systems engineers do, as well as how and why they do it. Being able to do it by the book is not enough. The Systems Engineering Body of Knowledge (SEBoK) (Faulconbridge and Ryan, 1999) could be a start in meeting these requirements.

7.2.2 Experience requirements

Years of experience is not enough, the applicant must show continual growth and success, perhaps by publications and presentations at INCOSE symposia, conferences and Chapter meetings. There must also be requirements for leadership experience and demonstration of successful project completion (Kasser, 1995).

7.2.3 Achieving compliance to the requirements for certification

Two models can be adapted. One approach requires that the applicant sit a comprehensive examination written by the administering organization. However, writing suitable questions is not a trivial task. They need to be multiple choice to simplify grading the responses, but there is a skill in writing good questions. The question has to be phrased in such a manner that the answer is obvious to someone who understands the subject while at the same time two or more of the solutions seems correct to someone who does not know the subject[28].

Another approach would be based on achieving a master's degree in a discipline covering the SEBoK.

7.2.4 Administration

This is the hardest issue because the administering body has to have a degree of credibility in the discipline. INCOSE is an organization in which anyone interested in the topic can pay the fee and join. This is the same as the Institute of Electrical and Electronics Engineers (IEEE). Membership does not provide an indication that the member has achieved a threshold of knowledge. On the other hand, the IEE has particular education requirements for degrees of membership

These organizations are not suitable as developers and administrators for several reasons that include:

[28] Based on a seminar given by the Educational Testing Service and experience in the workshop that wrote the questions for the initial Certified Quality Manager examination for the ASQ.

- They are not suited for developing and prototyping a certification process in a short period of time.
- They don't have the understanding of the SEBoK.

The Systems Engineering Society of Australia (SESA) with its links to INCOSE and IEEAust seems to be a neutral body with the qualifications and expertise to develop and administer the prototype certification process. Its location in Australia, these days is no impediment. While it takes time to move people and packages by air, electronic communications are essentially as fast between Australia and the rest of the world as they are between any two points within a hundred miles of each other in the US or in the European Union.

The recommendation is that SESA be the administering body in its prototyping phase.

7.2.5 Recertification

To ensure that systems engineers remain current with emerging techniques, periodic recertification as employed by the ICPM is required.

7.3 Levels of certification

Recognizing that systems engineers are grown there should be master and apprentice levels of certification. The ICPM uses a two level approach for certifying managers. In systems engineering the recommendation is for two levels:

- **Certified Systems Engineer** – someone who has demonstrated compliance with both the educational and experience requirements.
- **Associate Certified Systems Engineer** - someone who has demonstrated compliance with either the educational or the experience requirements.

7.4 The prototyping approach

It could take years to set up the perfect certification program. Getting all the stakeholders to agree on the requirements and if specific applications meet them is a major problem. The proposed solution to the setting up problem is to attempt to avoid it by using the "Rapid Prototyping" approach to set up a baseline, start doing the job, then measure the results and modify the process using change control techniques. The proposed sequence is:

1. Develop a baseline version of the SEBoK.
2. Develop the experience requirements.
3. Publish the requirements for certification.
4. Begin certifying systems engineers based on academic qualifications by the year 2002.

5. Evaluate the program after 12 months.
6. Upgrade the certification requirements every 24 months.
7. Develop an examination over the next 12 months.

7.5 Conclusions

There is a need for certification of systems engineers. Obtaining consensus on the requirements for certification is not a trivial problem and will take time and lots of discussions. A prototyping approach was suggested as a way to minimize or avoid the consensus problem. SESA is an ideal organization to prototype the certification process. It is time to go for it.

7.6 Summary

This Chapter looked at improving systems engineering from the perspective of the people who perform it, namely the systems engineers. Recognizing that a process is only as good as the people who perform it, this Chapter defined the requirement for the certification of systems engineers to establish a minimum level of competency for systems engineers. After a brief survey of how other disciplines certify their practitioners, this Chapter discussed the derived requirements for implementing the certification process and recommended an approach to prototype and administer the certification process for systems engineers.

2000

8 A framework for requirements engineering

This Chapter views the SDLC from the perspective of a production system:

- focusing on the cost of both the process and the product;
- introducing the Framework for Requirements Engineering in a Digital Integrated Environment (FREDIE) which is an object-oriented tool arising from the Anticipatory Testing concept (Kasser, 1995) with the potential to implement significant cost reductions in the SDLC;
- outlining how those cost reductions can be achieved; and
- presenting some Use Cases of the FREDIE tool.

The SDLC has evolved several methodologies since the early days of the Waterfall model (Royce, 1970). One of them, the Spiral model (Boehm, 1988) pages 61-72) is the waterfall modified to place explicit emphasis on Risk Management. However, even with Risk Management and the current emphasis on Process Standards and Capability Maturity Measurement, the developer working within the current production paradigm, cannot answer two simple questions posed by the customer during the SDLC, namely:

- "What Do You Mean, You Can't Tell Me How Much of My Project Has Been Completed?" (Chapter 4).
- "What Do You Mean You Can't Tell Me if My Project is in Trouble?" (Chapter 5).

There has been a lot of research into building the right system and doing requirements better (Glass, 1992). Much of that research has focused on how to state the requirements in the form of a specification once they have been obtained, using a requirements traceability matrix (RTM), and the tools that incorporate a RTM. Consequently, while the implementation of good system and software requirements management practices is believed to be one of the first process improvement steps an organization should take, implementation still remains a challenging problem (El_Emam and Hoeltje, 1997). One solution to this problem is Anticipatory Testing (Kasser, 1995).

8.1 Anticipatory testing

Anticipatory Testing combines prevention with testing and is based on the recognition that prevention is planned anticipation (Crosby, 1981) page 131). The Anticipatory Testing approach concept is a control and information system paradigm rather than a production paradigm. It views the SDLC from the perspective of Information Systems, the application of Knowledge Management and modern Quality theory. It has explicit emphasis on Configuration Management and building Quality into the process. From the Anticipatory Testing perspective, a requirement can be thought of as consisting of two parts; the functionality and the Quality criteria that define the measurable attributes associated with the functionality. The term Quality is used based on the definitions of Quality as "*conformance to specifications*" (Crosby, 1979) page 131) and as *"fitness for use"* (Juran, 1988) page 11). For example, a requirement to ingest sensor data into a system is made up of the function that ingests the data and the minimum measurable amount of data to be ingested over a specified period of time within a specified data error rate. Having to consider both the functionality and Quality criteria components of a requirement should tend to ensure that requirements are quantifiable and hence verifiable at the time of acceptance. Then later when the requirement is decomposed into subsystem requirements the flow down of quantifiable subsystem requirements should also tend to ensure that the capability provided by the system meets the performance desired by the customer.

Anticipatory Testing is used within an Organizational Engineering or integrated product-process and management paradigm (Kasser, 1999). The most significant factor in the Anticipatory Testing approach is the recognition that cost reductions (improvements) in the product and process do not occur in a vacuum (Kasser, 1995). The product under construction is a system and the process producing the product is a system. People working within the context of an enterprise framework (system) build a product over a period of time. Thus, the process, product and organization represent three tightly coupled dimensions of the system and must not be considered independently. In addition, every one of the systems changes over time[29]. From the Anticipatory Testing perspective the [model] **SDLC is a time-ordered sequence of activities and can be considered as a series-parallel set of phased Builds (mini waterfalls or cataracts) in a multithreaded environment under the control of the Configuration Control Board (CCB)[30].** Figure 8-1 presents this concept by showing the traditional Waterfall meth-

[29] This concept was the basis for the PPPT model shown in Figure 3-2.
[30] See the cataract methodology in section 13.

odology sequential elements connected via a CCB that allocates the implementation of requirements and subsequent changes to Builds in which:

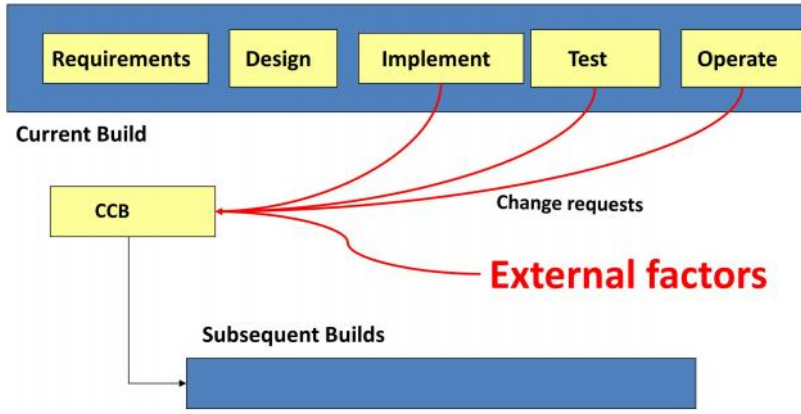


Figure 8-1 Anticipatory testing view of the SDLC

- **Engineering** converts the real user needs into capability (functionality and Quality criteria (requirements)) and arranges functionality into sets (Builds) namely the design process.
- **Management** ensures that Builds are implemented in a phased manner.

8.2 Change management

From the Information flow perspective, the conceptual process of accepting prospective requirements (before the baseline is set) contains the following steps summarized in Figure 8-2.

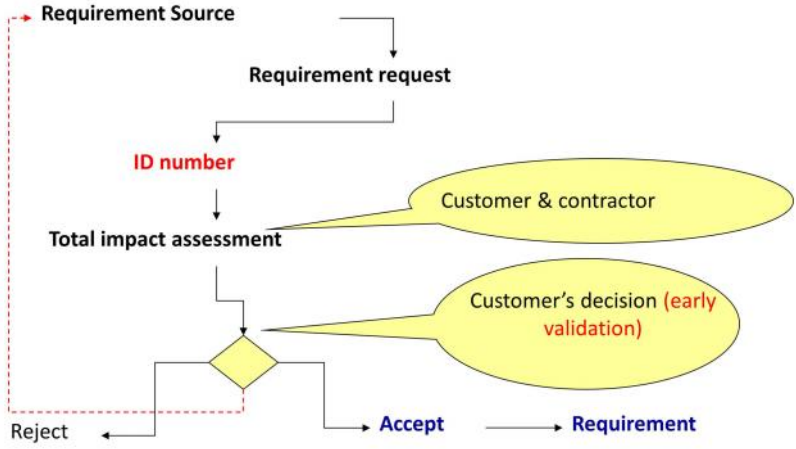


Figure 8-2 Conceptual Process for accepting requirements

1. Identify a requirement (based on a traceable source need or regulation) in the form of a Requirement request
2. Assign an identification (ID) number to the Requirement request.
3. Prioritise the requirement with respect to the other requirements.
4. Determine if a contradiction exists with existing accepted requirements.
5. Perform an impact assessment using an Integrated Product and Process Team (IPPT). The impact assessment must:
    - Estimate the cost/schedule to implement the requirement.
    - Determine the cost/schedule drivers – the factors that are responsible for the greatest part of the cost/schedule implementing the requirement.
    - Perform a sensitivity analysis on the cost/schedule drivers.
    - Determine if the high cost drivers are really necessary and how much negotiating the requirement with customers can make modification based on the results of the sensitivity analysis.
    - Make the decision to accept, accept with modifications, or reject the request.
    - Notify the originator.
    - Document the decision(s) in the requirement repository.
6. If the requirement is accepted, allocate the implementation to a specific future Build modifying the Build Plan and the WBS appropriately.

However, in order to perform the impact assessment and make informed decisions at any specific time in the SDLC in an effective manner, a certain amount of information is needed about the process as well as the user's need. In the existing production paradigm, much of this information tends to be lacking or if present is contained in several different and usually unconnected tools such as Requirements Management, Project Management, a WBS, Configuration Control, and Cost Estimation, etc. This information, herein named Quality System Elements (QSE) includes but is not limited to:

- **The Requirement** - the imperative statement containing both the required functionality and its corresponding Quality criteria or other form of representation.
- Unique identification number - the key to tracking.
- **Traceability to source(s)** - the previous level in the production sequence.
- **Acceptance criteria** – the answer the question "how will we know the requirement has been met?"

- **Priority** - knowing the priority allows the high priority items to be assigned to early Builds, and simplifies the analysis of the effect of budget cuts.
- **Risk** - any risk factors associated with the requirement.
- **Traceability to implementation** - the next level in the production sequence. Thus in software, requirements are linked to design elements, which are linked to code elements, and so on.
- **Estimated cost and schedule** - these feed into the management plan and are refined as the project passes through the SDLC.
- **The level of confidence in the cost and schedule estimates** - these should improve as the project passes through the SDLC.
- **Rationale for requirement** - the extrinsic information and other reasons for the requirement.
- **Planned verification methodology(s)** - developing this at the same time as the acceptance criteria avoids accepting requirements that are either impossible to verify or too expensive to verify.
- **Keywords** - allow for searches through the database when assessing the impact of changes.
- **Production parameters** - the WBS elements in the Builds in which the requirements are scheduled to be implemented.
- **Testing parameters** - the Test Plans and Procedures in which compliance to the requirements are scheduled to be verified.
- **Traceability sideways to document duplicate links** - required when applying the QSE to an existing paper-based project.
- **Access control parameters –** national security classification or company confidential as appropriate.
- **Version control –** for use in tracking changes. This is the version number, copies of older versions of the requirement and links to the CCB change database.

8.3 The FREDIE paradigm

Requirements Engineering is a discipline that is evolving from its traditional role as a mere front-end to the systems lifecycle towards a central focus of change management in system-intensive organizations (Jarke, 1996). For example:

- Dorfman and Thayer stated the definition of requirements engineering as "*the science and discipline concerned with analysing and documenting requirements*" (Dorfman and Thayer, 1990).
- Kotonya and Summerville restated the definition of requirements engineering as "*the systematic process of eliciting, understanding, analysing, documenting and managing requirements*" (Kotonya and Summerville, 2000).

This Chapter proposes the next stage in the evolution of Requirements Engineering by expanding the traditional RTM into an object-oriented database represented by the set of QSE to be stored in a FREDIE instead of in the several separate engineering and management tools currently in use. By requiring a full set of QSE for each requirement at the time the requirement is agreed to by the customer and contractor, some quality is built into the structure of a project. For example:

- The cost and schedule impact of a requirement or a change is known (to some extent) up front.
- The impact of change requests on the project can be more easily identified than in the paper-based production paradigm.
- The rational for the requirement, the acceptance criteria and the verification methodology are documented early in the process minimize:
- The ambiguity in poorly written requirements
- The imposition of unrealistic and unverifiable requirements.

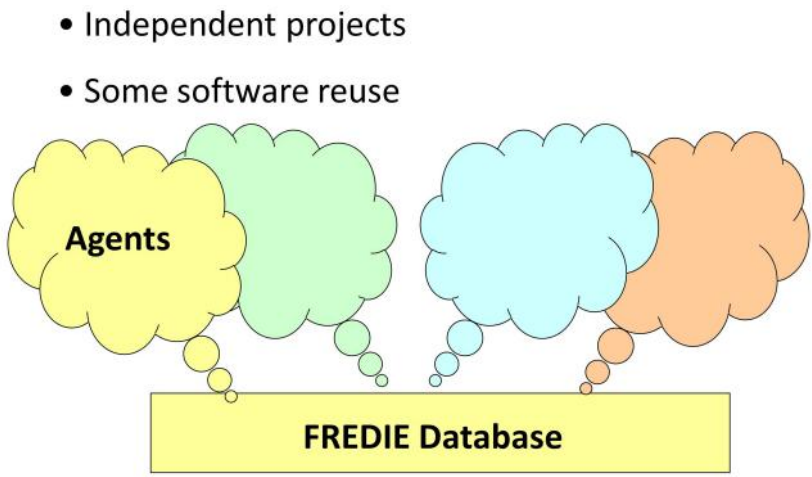


Figure 8-3 The FREDIE concept

The FREDIE agents[31] provide an integrated digital environment and are built as described in (Kasser and Cook, 2003). Access to the QSE via the FREDIE Agents as shown in Figure 8-3 will allow decisions to be made more rapidly and effectively. Thus the concept will improve the shared meaning as well as El Eman and Madhavi's three dimensions of Requirements Engineering success[32] (El_Emam and Madhavji, 1996):

[31] Software acting upon the QSE data in the FREDIE.
[32] Note the similarity to the Quality-Index.

- Cost effectiveness of the process.
- Quality of the Requirements Engineering products.
- Quality of the Requirements Engineering service.

8.4 The value of a FREDIE

The value of a FREDIE can best be implied by demonstrating its use in various scenarios as shown herein.

8.4.1 The use of acceptance criteria to identify the customer's real needs

Acceptance criteria are important properties that not only drive the testing stream of work; they also facilitate building the right system in the first place. Someone pays for the defects to be produced, then pays for the repairs (Deming, 1986).

8.4.2 Change request - impact assessment

The conceptual process for handling a change request is shown in Figure 8-2. A source generates a change request, which is logged and assigned an identification number. Once the specific WBS element[33] and/or requirement (as appropriate) affected by the change are identified, then the links in the FREDIE facilitate an informed assessment of the impact of the change on the other elements (capability, cost, schedule, risk, WBS) within the project. The impact of the requested change on the product and process (Builds) is assessed and a decision made as to whether to accept or reject the request. The source is then notified of the decision, if the change request is accepted, then, if the configuration control process is fully operational:

- From the product perspective, the affected requirements and all subsequent project documentation must be changed to reflect the new situation. This is done by adding, deleting or modifying (a combination of adding and deleting) requirements. The change may affect the capability of components at various levels of the design.
- From the process perspective, the Build Plan must be changed to show when and where the change will be implemented by changing the affected elements of the WBS. The cost and schedule impact will then be seen.
- The Systems Engineering Management Plan (SEMP) and Operations Concept Document (OCD) must be modified as appropriate.

[33] Implementing the requirement.

However, in most current instances the configuration control process is defective and one or more of the steps listed above do not take place.

8.4.3 Project quality audit

If the FREDIE database is populated by the requirements, configuration control and project management data for a project, a Quality Audit may be performed. The audit is performed by examining the data in the FREDIE database. With the addition of the appropriate knowledge base for the specific function, this audit may perform several functions including:

- Identification of some poorly written requirements by scanning the text of the requirements for words that do not meet the requirements for writing requirements (Kasser, 1995; 2002c).
- Identifying work that is not being done, or work that doesn't have to be done, by finding missing links in the traceability of the requirements to the WBS.
- Identifying missing links in the traceability of the requirements to test plans and procedures.
- Identifying missing activities or steps in the processes within the SDLC.
- Identifying other missing information.

8.4.4 Categorized Requirements in Process (CRIP)

The CRIP Chart technique discussed in Section 5.2 is a way to estimate the percentage of project completion by looking at the change in the state of the requirements over time from several perspectives. Implementing the CRIP Chart technique using a FREDIE could help build "prevention" into the tool. It would be straightforward to build intelligent software agents to automate problem identification and assist human interpretation of the more subtle trends identified by FREDIE agents as outlined for the PERCY (Chapter 3).

8.5 Summary

Viewing the SDLC from an information system paradigm rather than the current production process paradigm has the potential to provide a significant reduction in the number of project failures and overruns by building the concepts of Quality and configuration management into the project tools. The FREDIE tool approach is an innovative change in the continuing evolution of the tool set used for requirements engineering and management.

2000

9 Enhancing the role of test and evaluation in the acquisition process to increase the probability of the delivery of equipment that meets the needs of the users

This Chapter views the organisation from the perspective of T&E which is becoming increasingly important in current acquisition practices as a way to ensure that the military equipment user receives equipment that conforms to its requirements. Now, T&E really has two different roles in which:

- **Testing** – determines the degree of non-conformance to requirements of "as-delivered" equipment (does the equipment do what it is supposed to do?).
- **Evaluation** - determines the capability (functionality and performance) of "as-delivered" or "as-built" equipment (what can the equipment actuality do?).

In recent times, recognizing that the documented requirements do not generally represent the true needs of the user, the T&E role has expanded itself to attempt to ensure that "as-delivered" equipment meets the needs of the user. This Chapter discusses the reasons for, the differences between, and how modern Quality theory, Information Technology and Knowledge Management, can improve the various roles of T&E.

The traditional SDLC is characterized by large cost overruns, schedule slips, and dramatic performance deficiencies in weapon, and automated information systems. This situation can be depicted in the view of the SDLC from the front end shown in Figure 9-1. The Figure represents s that the system requirements are not only incomplete and poorly articulated, they

are also evolving faster than the contractor can build the system. This is a different way of drawing the situation shown in Figure 5-2 and providing a little more information pertinent to this discussion[34].

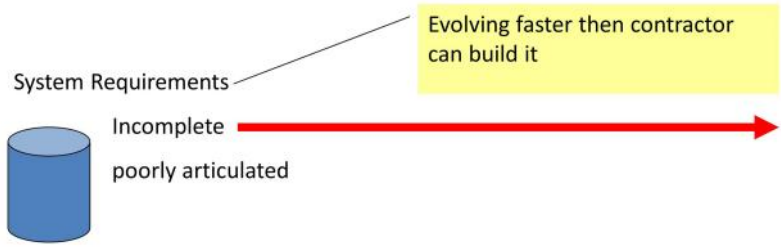


Figure 9-1 View of the SLDC from the front end

As a consequence, there have been a number of reactions to this situation; including the development of the CMM and the requirement for conformance to process standards. A less-publicized, but important reaction has been the expansion of the role of T&E.

9.1 The expansion of T&E

The view of the SDLC from the T&E and IV&V perspective which tends to get involved in the program after the requirements have been frozen is shown in Figure 9-2. T&E tends to see the requirements as poor, evolving, and incomplete, and has to use their domain expertise of the application to do their best to ensure that the equipment is fit for use. Dedicated T&E practitioners in the Defence world, recognizing that the "as-delivered" military equipment does not meet the real needs of the users, have expanded the role of T&E to ensure that military equipment fielded in the Defence forces (as-delivered) is suitable for use before being placed into service. The drivers for this expansion of T&E include (Dennison, 2000):

- Increasing complexity of aircraft and systems.
- Longer development programmes.
- Longer periods of service.
- Changes in policy on airworthiness and liability.
- Funding pressures.

[34] Figure 9-1 and Figure 5-2 present slightly different views of the SDLC abstracting out the pertinent information that is relevant to the situation being discussed in association with the Figure. This is an example of Simplicity as described in Chapter 18.

From the systems engineering perspective, these drivers may be summarized as:

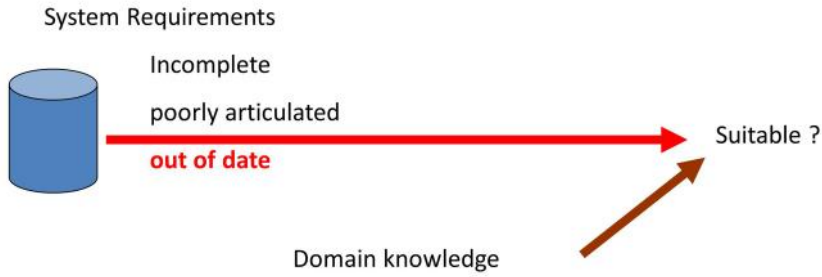


Figure 9-2 View of the SDLC from the T&E perspective

- Equipment is being built to specifications that are poorly written, ill defined, and incomplete.
- The need to manage changing requirements over the development and Sustainment phases of the SLC.

9.2 T&E in the United States Air Force

In 2000, the US Air Force (USAF) T&E procurement of weapon systems was divided into two roles (Pearson, 2000):

- **Developmental test and evaluation (DT&E):** the USAF uses DT&E to learn and confirm, that is, to learn about the system's capabilities, and confirm that it performs according to specifications.
- **Operational test and evaluation (OT&E):** the USAF uses OT&E to answer two fundamental questions.

    1. Given a realistic environment, can the warfighter use the system to accomplish the mission?
    2. Given the same realistic environment, can the warfighter support and maintain the system?

The USAF makes this clear distinction between DT&E and OT&E to recognize the fact that while a weapons system may meet all the design specifications, it may still fail to accomplish the mission.

Now from the perspective of modern Quality theory, finding defects after production is complete is not very cost-effective. This is because the customer pays for the defects to be produced, then pays for the contractor to affect the repairs (Deming, 1986). Thus the USAF T&E approach does not embody modern Quality theory and only mitigates the symptoms. It does not remove the root cause of the problem in the system namely the failure to ensure that effort be expended to ensure that the mission requirements are incorporated in the design specifications. This is the function of systems

engineering since the goal of the system engineering effort (Kasser, 2000c) is to provide a system that:

- Meets the customer's requirements as stated when the project starts.
- Meets the customer's requirements as they exist when the project is delivered.
- Is flexible enough to allow cost effective modifications as the customer's requirements continue to evolve during the operations and maintenance phase of the system lifecycle.

Meeting this goal on large systems developed over a period of years is practically impossible with today's technology. The current acquisition scenario, which takes place within the context of a production paradigm, is characterized by poor requirements and contains poor management of change (Chapter 5). Now the universe of requirements embodies a number of categories for Capability, namely:

1. Capability that is desired.
2. Capability mandated by external constraints liable to change, such as Government regulations, etc.
3. Capability mandated by external constraints that are unlikely to change, such as the laws of physics, etc.
4. Capability that does not matter to the user one-way or the other, and the development contractor is notified of that situation.
5. Capability that does not matter to the user one-way or the other, and the development contractor is not notified of that situation.
6. Capability that is not desired.
7. Capability that is desired but the customer does not know that it can be provided.
8. Capability that is desired but cannot be provided.
9. Capability that is irrelevant to the equipment to be acquired.

The full set of user requirements for a system to be acquired tends to be embodied in the first four of the categories.

9.3 Enhancing the traditional role of T&E

The traditional role of T&E is to ensure that the product meets its real requirements. In order to perform this role as early as possible in the SDLC, T&E has to ensure that the correct system is built in the first place. It is supposed to do this by determining if the set of user requirements produced by systems engineering are complete, verifiable and understandable. Verifiability and understandability may be improved by ensuring that the format of the written requirements conforms to the requirements for writing good requirements (Kasser, 1995) and by developing a metric for the goodness of requirements (Kasser, et al., 2006). Completeness, however, has been more

difficult to achieve. In any event even verifiable and understandable requirements change over the SDLC and the effects of change need to be managed over the SDLC.

T&E has not been the only engineering area of activity to recognize the existence of poor requirements engineering management. Requirements engineering itself is evolving from its traditional role as a mere front-end to the SDLC towards a central focus of change management in system-intensive organizations (Jarke, 1996). Evidence of this evolution can be seen in the changes in the definition of the term requirements engineering quoted in Section 8.3.

Both systems engineering and T&E can make use of this concept in an interdependent manner by using modern Quality theory, Information Technology and Knowledge Management techniques to expand the traditional RTM into the QSE stored in an object-oriented database in a FREDIE tool (Chapter 6). The format of the QSE data set is such that each user requirement accepted for the deliverable product by the contractor must be accompanied by both specific measures to determine when compliance has been achieved, and a verification plan. This "forces" the customers to think about how they will know that the requirement has been met at the time the requirement is stated. So in the early phase of the SDLC, while systems engineering are developing the requirements and implementation cost and schedule estimates, T&E are clarifying the requirement through the use of acceptance criteria and devising the verification methodology and the cost and schedule estimates for the testing effort as shown in Figure 9-3. When the domain knowledge goes into the systems requirements up front, there is a greater probability that the system will function in the domain.

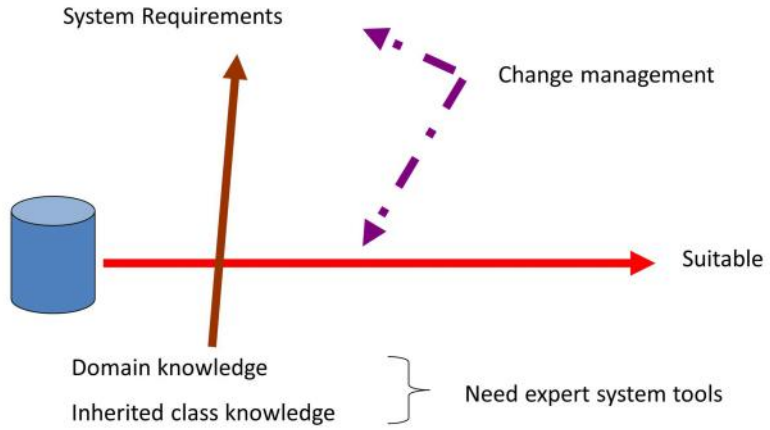


Figure 9-3 Building the right system

9.4 How T&E can reduce some categories of missing requirements

Traditional systems engineering has focused on performance and functional requirements as shown in Figure 9-4. The requirements for the mission are document in the System Requirements Document (SRD) together with the known enablers and constraints. This process often results in the omission of critical non-performance and hence in the delivery of systems that are not fit for use.

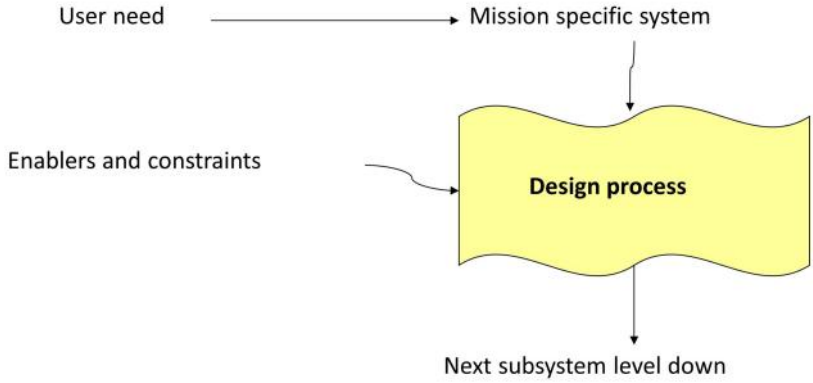


Figure 9-4 Traditional requirements flow down

To maximize the completeness of requirements and reduce the effect of non-mission-specific missing requirements, both systems engineering and T&E must also ensure that system to be developed inherits generic requirements applicable to the specific type of product. For example, a low earth orbiting (LEO) spacecraft should inherit a set of generic requirements that have been refined from the experience gained in building these types of vehicles over the last fifty years. These requirements relate to the thermal, vacuum, and electromagnetic environment in space, launch, vibration, and salt spray, etc. Should the LEO spacecraft be a communications satellite, there would then be an additional set of generic requirements to be inherited. As a second example, equipment for use in jungle environments should inherit a set of temperature and humidity requirements while the same or similar equipment destined for use in a desert environment will inherit a different set of temperature and sand-resistance requirements. These generic sets of inherited requirements, stored in knowledge bases, can ensure that important but tangential non-functional requirements are not forgotten in the design phase, e.g. ensuring that equipment that has to be shipped actually fits through doors and air cargo containers. The addition of these mission generic requirements is shown in Figure 9-5.

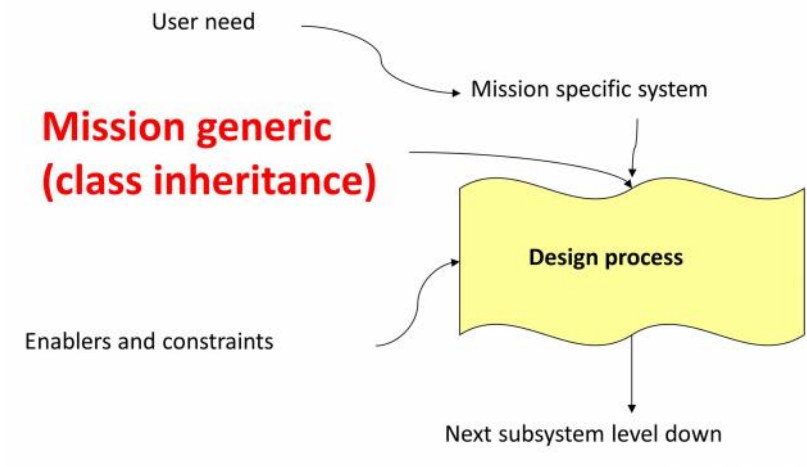


Figure 9-5 Mission generic requirements

These generic sets of potential inherited requirements currently tend to be based on the experience and education of the practitioners. If the project has the expertise then those requirements are considered and incorporated in the SRD. Should the expertise be lacking, they may be missed and have to be incorporated further down the schedule necessitating rework and thus contributing to cost escalations, and schedule delays. These requirements or mission specific knowledge can be available in generic knowledge bases, which then provide a set that must be tailored for each specific project. T&E by virtue of being able to build a lesson's learned database from the test results of past projects is eminently suitable for providing much of the input to populating the generic knowledge bases for various categories of systems. For any specific new project, the roles of systems engineering and T&E will depend on the project, but the generic functions might be:

- T&E provide the full generic set of requirements based on a whole range of past projects in a knowledge base to reduce the number of missing requirements. This is a corporate (support) activity.
- Systems engineering tailors the generic set for the project by importing them into the project's FREDIE tool. This is a project (production) activity.
- T&E verify that the tailoring performed by systems engineering conformed to the correct pattern for the system. This is a project (production) activity.

Once the initial set of requirements has been accepted, experience has shown that they will change over the SDLC. Efficient through-life systems engineering, change management, and T&E, will require a planned, coordinated and integrated approach as well as data/knowledge warehousing

(Dennison, 2000). The FREDIE tool can facilitate change management by virtue of its object-oriented data structure and its ability to use Knowledge Management techniques in the form of smart agents. It is thus advisable if not incumbent to ensure that a FREDIE tool is used for new projects to avoid much of the costs now incurred in post-production rework and retest, to facilitate managing requirements, design, development, integration, and T&E, in a cost-effective concurrent manner.

9.5 Determining the capability of "as-delivered" equipment

Another major role of T&E is to determine the capability of "as-delivered" equipment. This is where Evaluation determines the:

1. Extent, by which the requirements have or have not been exceeded.
2. Unanticipated emergent properties of the system that were not predicted based on the functionality of the individual subsystems.
3. Suitability (performance) of previously owned equipment, commercial off the shelf (COTS) or equipment built for another customer and made available at a price or schedule that is less than that of custom produced equipment.

9.6 Requirements and capability

Think of a requirement as consisting of both functionality and Quality criteria. For example, a requirement to ingest sensor data into a system is made up of the input function and the performance parameter for how much data to ingest over a specific period of time. Capability can also be made up of functionality and Quality criteria (what is done and how well it is done). Thus "as-delivered" equipment has four possible states:

1. Capability is exactly equal to the requirements: the product meets its requirements.
2. Capability exceeds requirements: either the performance parameters exceed requirements, or un-required functionality is delivered.
3. Capability does not meet requirements: either the performance parameters do not meet requirements, and/or required functionality is not delivered.
4. A combination of the previous three.

A comparison of the full set of user requirements and the capability (expressed in terms of functions and Quality criteria) for a typical small system can be represented in Kiviat or Bar Chart format as shown in Figure 9-6. The chart to use is the one familiar to the viewers.

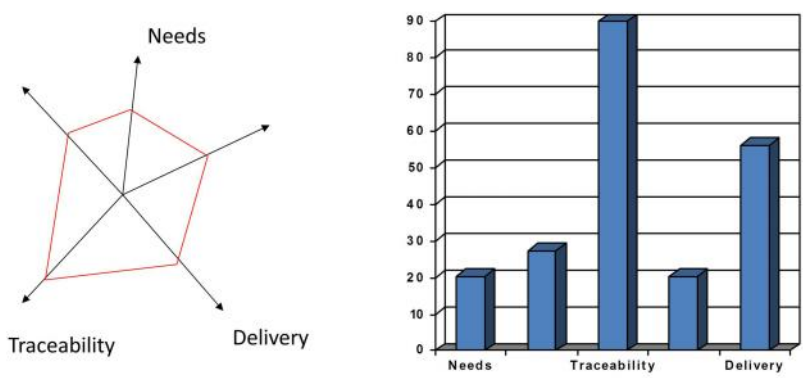


Figure 9-6 Kiviat and bar chart comparison

If Evaluation (1) is performed on behalf of the contractor, it may be used to identify places where costs might be cut by reducing unneeded performance (system exceeds requirements). This information is important in a fixed-price mass-production environment, for example in the automobile construction industry a per-item saving of $1 can be significant on a production run of 500,000 units.

If Evaluation (1) is performed on behalf of the customer it may be used to identify additional capability. For example, supposing an aircraft specified to perform a turn at 2 G's under certain circumstances is found to be capable of a 4-G turn under the same circumstances. This additional performance might allow the pilots to develop a new manoeuvre. The importance of this role of T&E is that it provides information about the additional capability of the equipment which then allows the user to develop additional missions or uses that may not have been present in the original concept of operations for the equipment.

9.7 Software T&E

The last 30 years have seen a transition from hardware-based systems to software intensive systems operating on hardware platforms. T&E has to ensure that the hardware-software combination is appropriate. Thus the:

- Evaluation (1) component of T&E needs to incorporate the evaluation of excess capability offered by the COTS components included in the software.
- Evaluation (2) component has been traditionally associated with physical characteristics such as temperature and electromagnetic compatibility issues. In today's software intensive systems the Evaluation (2) component will also need to address issues arising from the interaction of the multiple complex software-based subsystems.

9.8 Summary

T&E has several roles in the acquisition process. By:

- Identifying non-conformance to requirements, T&E prevents the delivery of unsuitable equipment to the end user.
- Identifying and reporting the capabilities of "as-delivered" equipment over and above the user requirements, T&E provides a valuable service to the user.
- Collecting generic requirements on types of systems and providing them to systems engineering, T&E can help to minimize more and more categories of missing requirements in future systems.

By making use of modern Quality theory, Knowledge Management and IT, T&E and IV&V are positioned to work with systems engineering to prevent defects, test for non-conformance to requirements, and evaluate the capability of "as-delivered" equipment in a cost-effective manner over the entire SDLC.

2000

10 Systems engineers are from Mars, software engineers are from Venus[35]

This Chapter views the organisation from the perspective of its culture – the people who comprise the organisation and how they communicate because communications is a vital ingredient to the success of any enterprise. Once work is split up between different people the need to communicate shows up. One of the many barriers to successful communications is the background of each individual which assigns different meanings to words. For example, consider the word "secure". When told to "secure" a building it has been related that,

- The Navy issues a purchase order for the building.
- The Air Force locks the doors and turns on the alarm systems.
- The Army evacuates the personnel, then locks the doors and turns on the alarm systems.
- The Marines assault the building using ground troops and air support, and then deploy squads in and around the building checking the credentials of all who aspire to enter the building.

This example illustrates a subtle communications problem. When one hears unknown words, such as in a foreign language, the failure to communicate is obvious. However, when one hears words that sound correct in the context, the failure to communicate is not realised and sometimes produces serious consequences. This situation can happen when communications takes place between different organisations, different national cultures and even different engineering specialties as demonstrated in this Chapter. The differences between systems and software engineers has been chosen as a demonstration, because in the last three decades of the 20th Century, a

[35] Acknowledgment is given to John Gray for the variation on his book title "Men are from Mars, Women are from Venus".

period roughly matching software engineering's history, the development of complex projects seems to have resulted in massive failures. This Chapter provides some insight that the failure of systems and software engineers to communicate, and more importantly their failure to understand that they are not communicating, may be a hitherto undetected cause of the failures experienced in developing today's complex projects. The Chapter then explores some of the reasons for the communications failure and recommends an approach for improving the ability of systems and software engineers to communicate.

Five thousand years of experience in hardware engineering have provided its practitioners with methodologies that have, in the main, been successful. It was during that time that projects were engineered. Only in the last three decades of the 20th Century, a period roughly matching software engineering's history, has the development of complex projects been plagued by massive failures. Yet when seen from a historical perspective the complex projects of the last three decades are no more complex than the large engineering projects of the past, projects like the railroads, canals, pyramids and military sieges, within the constraints of the then-available tools and technology[36]. The growing recognition that the multidisciplinary environment in which today's complex software intensive systems are developed is characterized by poor requirements, poor change management and poorly defined processes resulted in several approaches to improve software development, including:

- The development of Computer Aided Software Engineering (CASE) Tools.
- The focus on process and methodology embodied in the ISO/IEEE standards.
- The US DOD initiated Software CMM.
- The development of the "domain abstraction concept" in object-oriented Methodologies (OOM).
- The adoption of the UML.
- The development of reusable generic software components.
- The inclusion of software engineers on IPTs.

However the adoption of these approaches will not guarantee that current and future software based complex projects will not fail. This is because the projects that software engineers develop do not exist in a vacuum (Pfleeger, 1998). Software cannot perform its function without an underly-

[36] While we can see the products or outputs of those development processes, little information about cost and schedule overruns or prior failures has survived through the ages.

ing hardware platform. Consequently, software engineers need to communicate with the systems engineers, hardware engineers and the other members of the IPT.

During the course of some research Sharon Shoshani (the software engineer) and I (the systems engineer) show in in Figure 10-1 realized that a communications gulf existed between systems and software engineers, and tried to identify it. After some discussion[37] we developed a conceptual meta-model of the system development process within the SLC. The meta-model, which became our Rosetta stone, summarizes the development process in the following manner: When faced with the problem of meeting the customer's needs:

Figure 10-1 The systems and software engineers

- Engineers don't always reuse solutions or components that worked in the past; they reinvent them or try to invent new ones.
- Even when engineers try to reuse solutions or components that worked in the past, according to good engineering practice, there are two implementation choices:

    1. The problem is similar to other problems that have been solved in the past. Thus this time around, providing a similar solution may solve the problem. The process then becomes one of identifying the applicability of the solutions of the past, to the problem of the present and applying the elements of one or more solutions of the past to solve the problem of the present.

[37] No blood was shed in the process but we came close.

2. The problem is unique so there are no known solutions. The process then becomes one of identifying a solution that makes the maximum use of existing solutions to past problems (components) and the minimum use of components to be developed so as to reduce the risk of non-delivery on time and within budget.

At this time, while we came to the realization that while much of the communication problem was with the semantics; we also identified other underlying barriers including:

- Training and Background Differences
- A lack of respect for the other's profession.
- The use of language
- The role of systems engineer in the SLC.
- Different Concepts.

Before considering each of the barriers to communications it is important to distinguish between a true barrier, and poor application of the methodology since each engineering discipline has both good and bad practitioners. Each barrier in the above list is discussed in this Chapter and several examples are cited from the published literature as well as from personal experience.

10.1 Training and Background Differences

This section analyses the different training and background of software, systems, and hardware engineers.

10.1.1 Hardware Engineers

The most common background for system engineers is hardware engineering. Moreover, hardware engineering is often used as a reference when describing generic processes. Therefore it is important to understand the background of the hardware engineer. Hardware engineers have been using models and blueprints in the form of schematic diagrams and sketches since the early days of engineering. They also use a component-based methodology. However, they develop components that are physical in nature, e.g. black boxes, printed circuit boards, integrated circuits and subsystems. When hardware engineers design a digital system, they use integrated circuits. The hardware engineer will pattern match sections of the design to the offerings in the vendor's family of components. They then partition the design to make use of standard components and design a small amount of custom circuitry as required to interface between the standard components. The physical nature of the components themselves enforces the mapping of functionality onto physical components.

10.1.2 Software Engineers

In its early days software development practitioners used two basic approaches. Some developed their own libraries of subroutines or modules for functions, that once debugged, could be used in other programs. The majority however:

- Tried to implement the same functions in various ways in different projects.
- Suffered from the "not invented here" syndrome and would not reuse code from external sources.
- Failed to obtain the benefits of code reuse because they either changed working code or reused the code in a context different from its original one and the difference was the cause of the failure.

However, software engineers have since matured and created several more important methodologies, including

- **Componentware** - Components are large grained entities that are defined by their interfaces (services provided and demanded) and can be independently developed and used (Szyperski, 1997). The component approach promises to turn software development to software assembly in a similar manner to the use of integrated circuits by the hardware engineer.
- **Design Patterns** – a way of expressing general solutions to recurring problems.
- **Abstract modelling** – a systematic way of capturing the system requirements in an abstract, consistent and complete manner.

Software engineers are often in the vanguard when it comes to system changes due to:

- The common misconception that it is easier to make software changes than hardware changes.
- Software is responsible in most cases for the user interfaces – an area constantly requiring changes.

Software engineers are often a product of a three-year undergraduate course of computer and information science. This course seldom teaches math and physics at a level that is considered basic for engineering *("What do you mean you don't know what a Fourier Transform is*?" yelled the systems engineer at the software engineer.)

In a scan of the indices of books about software engineering and development picked at random from those used for, or considered for use for, teaching software engineering and management at the postgraduate level,

the following books identified the term 'systems engineering' in their indices[38] (Shumate and Keller, 1992; Pressman, 2005; Sommerville, 1998; Donaldson and Siegel, 1997; Brodie, 2001; Endres and Rombach, 2003). The remainder made no mention of the term (Shlaer and Mellor, 1988; Marcotty, 1991; Jacobson, et al., 1993; Arthur, 1993; Yourdon, 1993; Gamma, et al., 1995; Perry, 2000; DeMarco, 1995; Humphrey, 1995; Metzger and Boddie, 1996; Berg, et al., 1996; Yourdon and Argila, 1993; Budd, 1996; Fowler, 1997; Weinberg, 1998; Bass, et al., 1998; DeMarco and Lister, 1999; Van Vliet, 2000; Britton and Doake, 2003). Even Peters and Pedrycz made no mention of systems engineering in their index (Peters and Pedrycz, 2000). Thus a generation of software engineers seemingly was being taught to engineer software without knowing much about traditional engineering and what function systems engineers perform.

In addition, software engineering, which does not directly address systems engineering, is also teaching methodologies for eliciting requirements (e.g. Use Cases and business objects)[39]. At the same time it is ready to accept the system engineer as a requirement source in the manner of any other stakeholder.

10.1.3 System Engineers

Systems engineers in general have little if any formal training in systems engineering. They graduated to the discipline from another engineering discipline, mostly from hardware; therefore the software engineer claims that they may not have an understanding of the nature of software. In addition, software engineers are assuming the role of the systems engineer in software-intensive systems.

The systems engineering community has recognized the need for formal education and training in systems engineering (Chapter 7). A scan of the indices of books on systems engineering for the word "software" had mixed results. Software is cited in the index of several books on systems engineering (Rechtin, 1991; Thome, 1993; Eisner, 1997; Kendall and Kendall, 1997; Martin, 1997; Blanchard, 1998; Faulconbridge and Ryan, 2003). And although the requirements for software engineering are discussed in the context of adapting MIL-STD-2167A (MIL-STD-2167A, 1998) to systems engineering (Kasser, 1995), software only shows up in the bibliography section of Martin (Martin, 1997) and fails to show up completely in two books (Hitchins, 1992; Buede, 2000).

[38] Two of the books are used in the management classes not the engineering classes.
[39] This is in the B paradigm discussed in Chapter 28.

"Yes" pointed out the software engineer "while Buede doesn't list software engineering in its index, he borrows a lot of terms and concepts from software engineering" (Buede, 2000).

10.1.4 The lack of mutual respect

IPTs containing engineers from multiple disciplines develop today's complex projects. One of the characteristics of successful teams is respect for each other's specialty knowledge. Yet systems and software engineers, in general tend to have little respect for each other. This section explores the following reasons that system and software engineers fail to respect each other.

- The discipline of engineering
- Poor engineering practice
- The use of language

10.1.4.1 The discipline of engineering

Many of the software development personnel are not engineers but are programmers (equivalent to technicians). Other software engineers lack a formal background in math and physics, which tends to preclude them from fully understanding a system that has more than just software in it. On the other hand, many hardware and systems engineers have some programming experience, which makes them believe that they are "software experts". Comments heard in organizations included:

- "I tried using a real time operating system 15 years ago and it didn't work, therefore it shall not be used in my project".
- "This is something that my high school kid can program in a week".

10.1.4.2 Poor engineering practice

Some software engineers feel that systems engineers seldom provide the process-product deliveries expected from them. The software engineer claimed that she knew of several organizations that had to retrain their system engineers when going through the CMM process in order to be able to qualify for the CMM Level 2. Other common complaints were:

- "They keep handing us new requirements on the way to the canteen".
- "They fail to uncover all the system behaviour issues in their requirements, concentrating on the static and most apparent ones".

These examples of poor systems engineering lowered the respect for systems engineers in the eyes of the software engineers. On the other hand, from the systems engineer's perspective, Watts Humphrey created a CMM for the individual called the Personal Software Process (PSP) (Humphrey, 1995).

"His process, used by systems and hardware engineers for years, is being treated with wonder by the software engineering community because of the reception that the PSP is receiving" said the system engineer.

"Please don't mention the PSP," she retorted, "The PSP never was an issue in my community and was never actually mentioned or used".

"But listen to this" said the systems engineer quoting "*These techniques (the CMM, PSP, and others like them) apply basic engineering and management principles to software. Over the years, we software practitioners convinced ourselves that software was different. The basics did not apply to us; we needed to break the mould. We were wrong. Software is different in some ways, but it has more in common with other fields than we want to admit. We must use what others have proven works at work*" (Phillips, 1998).

In his eyes this quotation proves what systems engineering has held for a long time, namely, software engineering is one of many engineering disciplines.

The systems engineer then picked up a book on software reusability and read, "In well-established disciplines like civil or electrical engineering, reuse is based on the existence of previously coded knowledge. There are two different levels of reuse to consider: (1) the reuse of ideas or knowledge and (2) the reuse of particular artefacts and components. Electrical engineers, for example, consult component catalogues, check which available part best fits the design constraint, and, in some cases, relax the original design requirements to take advantage of existing components (Prieto-Díaz, 1987).

He, the systems engineer, agrees with the author but thinks that it is poor engineering practice to relax the original design requirements without consulting the customer. He has seen it happen in the world of systems engineering and knows that in the correct approach, the relaxation of requirements takes the form of a change request, then after the Configuration Control Board has accepted the request, a sensitivity analysis is performed and an informed decision made together with the customer. He reads on to discover

"We propose a model for reusability based on these observations and on the assumption that available components usually do not match the requirements perfectly, making adaptation the rule rather than the exception. Our approach is to provide an environment that helps locate components and that estimates the adaptation and conversion effort necessary for their reuse. The reuse process is as follows:

- *A set of functional specifications is given. The user then searches a library of available components to find the candidates that satisfy the specifications.*
- *If a component that satisfies all the specifications is available, reusing it becomes trivial".*

This is good stuff he thinks; software engineering has finally learnt to use components properly. He then reads on and discovers the following text:

"*More typically, several candidates exist, each satisfying some specifications. We call them similar components. In this case, the problem becomes one of selecting and ranking the available candidates based on how well they match the requirements and on the effort required to adapt the non-matching specifications. Once an ordered list of similar candidates is available, the re-user selects the easiest to reuse and adapts it*".

"Adaptation!" cries the systems engineer, "that's like performing the functional equivalent of drilling into an integrated circuit and attaching a wire internally, instead of designing some external circuitry to interface the wire to the component".

"No" replies the software engineer, "Adaptation does not necessarily mean changing or modifying the component. For example, when you bring a 110 Volt radio from the USA to Australia which uses 220 Volts how do you adapt it?"

"Well you can change the power supply internally, or add a transformer externally" was the reply.

"So the word adapt does not necessarily mean modify!" she said triumphantly.

"*Yes*" he admitted.

"So why did you immediately associate 'adapt' with 'modify'?" She asked.

"Because in my experience, that's what software engineers did, years ago," he replied.

After some discussion they agreed that indeed it was the perception that drove the reality, and even if today's software developers did not modify existing components, the systems engineer's perception, based on his experience, shaped his lack of respect for software engineers.

In addition systems engineers have the perception that software engineering much the same as any other engineering discipline is characterized by poor practice. For example, as the systems engineer said, Boehm states *"the software drives system considerations such as performance and cost. For example, in a recent survey of 16 books on object-oriented design, only six had the word 'performance' in their index, and only two had the word 'cost'"* (Boehm, 2000) .

"From the system engineering perspective, requirements drive performance and cost as well as functionality, which in turn drive the software and all other parts of the system. So why should Boehm in the year 2000 have to point out cost and performance to software engineering?" he asked.

She replied "You will not find cost in hardware engineering text books either, simply because the books discuss technology and not management. On the other hand I agree that in some areas of information systems technology performance is not treated with enough respect".

10.1.4.3 The use of language

Just as a common language opens the door to communication, so too the lack of it erects a barrier not easily overcome (Cox, 1996). However, even with a common language, communications is not guaranteed. While the notion that Great Britain and the United States are separated by a common language may be known (Shaw, 1925), its ramifications are subtle and it is not an easy concept to understand unless one has been sensitised to it. An example of the situation occurred in the mid 1970's during contract negotiations between the Communications Satellite Corporation (COMSAT) and British Aerospace. The language of the meeting was English. The meeting became stuck on one point. Someone then suggested "tabling the issue". Both sides agreed and the meeting deteriorated. The situation was much improved when the bi-lingual interpreter pointed out that the verb "to table" means in:

- **English** - to place the subject on top of the table for immediate discussion.
- **American** - to place the subject under the table or put it away for later discussion.

Consider the following SLC analogue, concerning the meaning of the word "component".

The systems engineer quoted "A component of a system is a subset of the physical realization (and the physical architecture) of the system to which a subset of the system's functions have been (will be) allocated" (Buede, 2000) page 50).

They agreed that systems engineers live in the physical world and the software engineer also stressed that components are not only physical but are also logical. So they decided to look up the UML 1.3 definition of a component and found "*A component is a physical, replaceable part of a system that packages implementation and provides the realization of a set of interfaces. A component represents a physical piece of implementation of a system, including software code (source, binary or executable) or equivalents such as scripts or command files*" (UML, 1999).

"So components ARE physical" said the systems engineer, to which she replied "What's physical for us is logical for you".

She explained further "software engineering separates the issue of component (what it does) from both deployment (where it executes) and from its instances (how many times it runs). It creates another layer of abstraction that has more in it than just the physical entity as thought of by systems engineering "

They then looked up the definition of a component in UML Version 2.0 and found the definition of a component as "*A modular unit with well-defined interfaces that is replaceable within its environment*" (UML, 2005).

"That means it can be physical or logical" she said.

"I think I'm getting a headache" he sighed.

10.2 The role of systems engineer in the SLC

"*Modern information technology products, even the software-intensive ones, are complex enough to require application of techniques from both disciplines. Accordingly, it becomes important to understand the relationships between relevant standards for systems and software engineering*" (Moore, 1998).

Singh depicts the link between system and software as: "*This standard establishes a strong link between a system and its software. It is based upon the general principles of systems engineering. The basic components of systems engineering (e.g., analysis, design, fabrication, evaluation, testing, integration, manufacturing, and storage/distribution) form the foundation for software engineering in the standard. This standard provides the minimum system context for software. Software is treated as an integral part of the total system and performs certain functions in that system. This is implemented by extracting the software requirements from the system requirements and design, producing the software, and integrating it into the system*" (Singh, 1995).

While mentioning systems engineering, the standard is silent as to the role of systems engineers. A search through engineering textbooks and other sources for the role of systems engineers showed that systems engineering is defined in several ways[40]. These different perceptions of systems engineering illustrate the point that systems engineers themselves have different perceptions as to their role in the SLC (Chapter 2). For example, the systems engineer pointed out that systems engineers might direct or perform:

- The high level design.
- Requirements and interface management.
- Activities that are not performed by other specialty disciplines.
- Inter-group coordination.
- The advocacy for the customer during the design and test phases of the task and whenever the customer is not present.

"In addition", he said, "in the 20th Century paradigm, which is based on building hardware, systems engineers convert customer needs to system requirements", and quoted from Chapter 2[41] "*Most of today's systems engineers really appear (work as) to be Requirements and Interface Engineers. They have the responsibility to validate the requirements since there's little*

[40] A representative sample of definitions is shown in Table 12-1.
[41] Actually it was the paper that became the Chapter.

point in building a system which conforms to requirements if the requirements are incorrect".

"In the 21st Century paradigm, the system engineer should lead the IPT". He added, to which she pointed out, "some modern software-intensive systems can be developed effectively without systems engineers as long as the software engineers perform the requirements elicitation function".

To which he replied "Pfleeger begins the process of describing a system by naming its parts and then identifying how the component parts are related to each other (Pfleeger, 1998). The system is analysed in terms of objects and activities. There is a role for requirements engineering to capture the system level requirements, but the book is silent as to how the requirements are allocated between the software and hardware subsystems".

10.3 The use of concepts

As an example of the different interpretations of concepts this section addresses the differences in the following concepts:

- Inheritance
- Blueprints
- Models
- Objects and classes
- Architecture
- Adaption
- Use of viewpoints

10.3.1 Inheritance

Systems and hardware engineers use inheritance in the form of physical and domain knowledge as shown in the following examples:

- When they build a printed circuit board they inherit mechanical aspects (size and shape) from other printed circuit boards. The board may also inherit connectors and interface circuits from previous boards.
- Spacecraft inherit environmental properties i.e. thermal vacuum, thermal, and vibration.
- Aircraft inherit attributes to make them airworthy.
- Ships inherit attributes to stop them from sinking and protect them from the long-term effects of seawater.

However they do not generally realize that they are employing the concept of inheritance, other than the obvious case of reuse, so they have no methodology for using the concept.

Software engineering uses inheritance as an integral part of object-oriented techniques. In modern software engineering you can inherit any classifier: interface, class, use case etc., thus creating a powerful way of ex-

pressing and implementing ideas for specialization and generalization, accompanied by rules of refinement.

10.3.2 Blueprints

"We have always used Blueprints in the form of engineering drawings" he said.

"We software engineer use blueprints as a metaphor claiming that software is always a blueprint even when implemented," she countered.

10.3.3 Models

"In my view a model is a representation of a product" he said. She replied "a model is a way of expressing knowledge in an abstract way, yet exact, without showing unnecessary details. We can use a model during analysis or design. We can use a model to generate implementation. Software engineers model as a way of communicating knowledge".

10.3.4 Objects and classes

In software engineering:

- An object is a discrete entity with a well-defined boundary and identity that encapsulates state and behaviour. An object is a role-centred entity with internal data and a set of operations provided for that data. An object is an instance of a class.
- A class is the type of an object. A class is a descriptor for a set of objects that share the same attributes, operations (methods), relationships and behaviour. Examples are a set of people, places, things or transactions that share common attributes and perform common functions (Dewitz, 1996). A class might capture real-world concept or design concept. A class can inherit another class's operation, relationship and behaviour either for expressing commonalties or as a way of making a more specific class. Objects can be instantiated from every class (not just from the more specialized one).

Objects and Classes originally were used in Object-Oriented Programming (OOP) to achieve encapsulation, reuse, inheritance and abstraction, later migrated to Object-Oriented Analysis (OOA) as a way of capturing the real world (the holistic approach).

In systems engineering an object is a subsystem. There is no concept of class. "*It goes back to the fact that you are actually not inheriting*" she pointed out.

10.3.5 Architecture

In software engineering an Architecture is "*The organizational structure and*

associated behaviour of a system. An architecture can be recursively decomposed into parts that interact through interfaces, relationships that connect parts, and constraints for assembling parts. Parts that interact through interfaces include classes, components and subsystems" (UML, 1999) or/and "*The set of design decisions about any system (or smaller component) that keeps its implementers and maintainers from exercising needless creativity*" (D'Souza and Willis, 1998).

In systems engineering, the architecture of a system consists of the structure(s) of its parts (including design-time, test-time, and run-time hardware and software parts), the nature and relevant externally visible properties of those parts (modules with interfaces, hardware units, and objects), and the relationships and constraints between them. There are a great many possibly interesting relationships.

"On the first reading they both look the same" she said, then added, "the software is adding a layer of abstraction that allows the developer to extend his problem dimensions".

"And later definitions include system evolution", he replied. "For example, the US DOD Architecture Framework (DODAF) provides the following definition of an Architecture – '*the structure of components, their relationships, and the principles and guidelines governing their design and evolution over time*'" (DoDAF, 2004).

10.3.6 Adaptation

The concept of Adaptation was addressed above in Section 10.1.

10.3.7 Use of viewpoints

Systems engineers' views tend to be constrained in the physical realm. So they tend to mix the physical and abstract views (Lykins, et al., 2000). This tends to result in a one-to-one mapping between the functional and the physical. Software engineers may pick any number of abstract views and try to separate concerns by splitting them to different yet related viewpoints.

10.4 Discussion

There are several reasons for the gulf separating software engineering from the hard engineering disciplines including:

- Systems engineers tend to think physically, and move rapidly to solutions. They do this using the reductionist approach of partitioning the big problem into a number of smaller problems on the assumption that if all the small problems are solved, then the big problem will also be solved. They also have a wide range of all sorts of components to assemble into their architectures (resistors, computers, tanks, aircraft, etc.)

- Hardware components generally evolved from one specific application to other applications. Systems and hardware engineers tend to focus on solving the problem, if they have to build a component; it tends to be a special purpose component optimised for that application. Hardware engineers constructed their own computer cards until vendor components become available. Similarly, companies developed proprietary network protocols until standards were adopted. Software engineers tend to think in abstractions and try to find an elegant solution to the problem (sometime staying abstract for a longer period than the system engineer feels comfortable with). In general, software does not have the benefit of thousands of years of component development. Thus when developing applications, software engineers may try to develop them in a manner that allows them be reused in the same or in other yet to be specified applications.
- A major barrier is the semantic barrier due to the different perceptions of the use of words. Both sides must be made aware that the situation is akin to Humpty Dumpty telling Alice that when he uses a word it means just what he chooses it to mean — neither more nor less (Carroll, 1872).

Consequently, in an exchange of information, each side should use active listening techniques to minimize loss of meaning across the gap in their common interface.

10.5 Bridging the communications gap between systems and software engineers

The key element in bridging the communications gap between systems and software engineers is to use active listening enhanced by "pattern matching" during discussions. For example, during a meeting discussing a scenario, design issue or requirement, they can often be seen to be similar to previous encounters. Thus when faced with a problem, the project team should first review this Chapter which will sensitise them to the issues and the potential barriers to communications. Only then should the discussions take place. During these discussions different people may recognize different similarities to previous encounters. Let each take a turn in explaining what pattern they recognize and why. Thus for example

- Systems engineering may recognize a Type A scenario.
- Hardware engineering may recognize a Type B situation.
- Software engineering may see it as a Type C.
- Reliability engineering may see it as a cross between Type B and Type C.

Participants in the meeting should use active listening techniques enhanced by pattern matching to apply feed back to the communications pro-

cess to maximize the probability of sharing the meaning. Active listening is a standard technique for applying the feedback principle to inter-personal communications to minimize errors in conveying the meaning from one person to another. Active listening first recognizes that during a conversation, most people do not listen to what the other person is saying. They are too busy planning what they will say when the other person pauses. Standard active listening comprises the following multi-step process:

When the other person speaks gives them your full attention and look them straight in the eyes. Then begin the following iteration loop.

1. Listen to everything the other person says and try to understand it fully.
2. Ask questions to clarify anything you don't understand and analyse the response.
3. Rephrase what you have heard in your own words and ask the speaker if they meant what you are about to say. Use words such as "if I understand you, then …", or "Do you mean…"? This is the principle of applying feedback.
4. If, after you have rephrased what has been said and the person says, "No that's not it!" or the equivalent, then go back to step 2. You may need to invoke the STALL technique at this time (see below).
5. When the speaker finally agrees with you, then you have (most probably) actually communicated and shared meaning.
6. In modifying active listening by the use of pattern matching, change Step 3 to incorporate the pattern by adding words such as "this reminds me of the [Type A Scenario]", and "isn't this similar to [Type B]" and explain why you find a similarity in the current situation. Use a metaphor appropriate to the other party such as sport.

- During the conversation use the STALL approach to regulate matters[42]. STALL is an acronym for
  - **S**tay calm
  - **T**hink
  - **A**sk questions and analyse
  - **L**isten
  - **L**isten

  You have two ears and one mouth, use them in that ratio.

[42] Stalling is a good way to initially deal with most problem situations.

10.6 Summary

This Chapter has provided some insight that the failure of systems and software engineers to communicate, and more importantly their failure to understand that they are not communicating, may be a hitherto undetected cause of the failures of today's complex projects. The Chapter then explored some of the reasons for the communications failure and recommended enhancing active listening with pattern matching as approach to bridging the communications gap. Active listening is a well-established technique for bridging communications problems and sharing meaning.

10.7 Conclusion

Systems and software engineers think differently, come from different backgrounds and use different vocabularies, hence stretching the point, it can be stated that systems engineers are from Mars and software engineers are from Venus.

2001

11 Requirements for flexible systems

This Chapter views the SDLC from the perspective of the product being produced and examines some attributes that make software and hardware systems flexible enough to be used for several purposes. Based on these attributes, the Chapter then develops some issues that need to be addressed in writing requirements for flexible systems. The Chapter concludes with lessons learned from the success, failure, and poor performance of systems that were designed to be flexible.

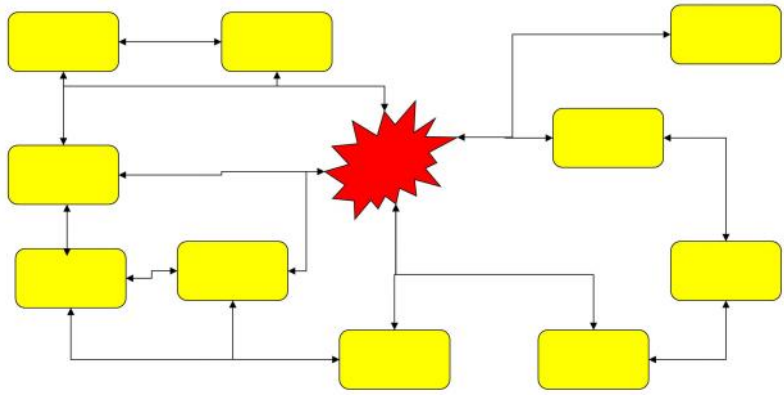


Figure 11-1 The context for the acquisition of a system

11.1 The context of a system

Systems in the acquisition phase have traditionally been considered as closed systems. Once the need had been established within a specific context, the context was rarely reconsidered during the remainder of the acquisition cycle. With the advent of the term Systems of Systems this closed system approach is transitioning into an open systems approach in which the big picture has to be considered when undertaking systems integration (Watts and Mar, 1997). So consider the system being acquired within the

context or framework of its adjacent systems. The context diagram is shown in Figure 11-1 where the system being acquired is the spiked object in the centre. The system has an interface with several other adjacent systems but not necessarily all the systems in the framework. The temporal perspective of the evolution of the same set of systems within the framework is shown in Figure 11-2. Each horizontal line represents an evolving system. The implementation and delivery of such systems and software are often performed in partial deliveries, commonly called "Builds" in which each successive Build provides additional capability[43]. Thus the sequential blocks in Figure 11-2 represent the operational phase of the different Builds within the individual system's SLC. The SDLC phases of the Build have been abstracted out to simplify the figure. Lines begin when new systems are brought into existence, and terminate when existing systems are decommissioned.

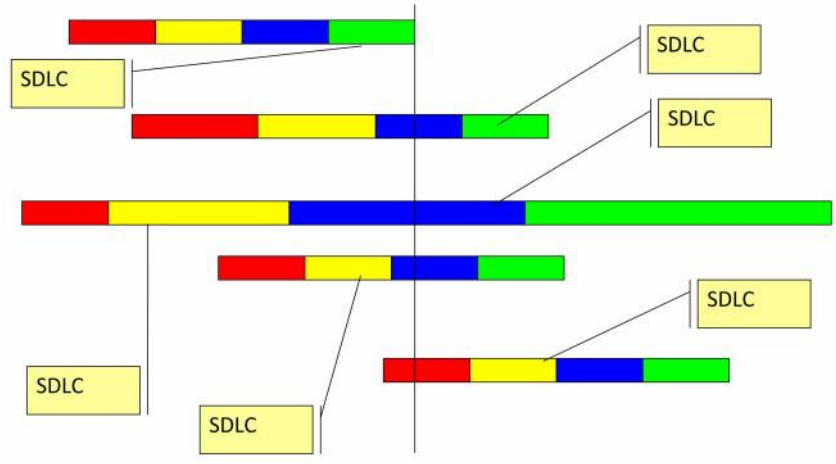


Figure 11-2 Evolution in context

11.2 The need for flexibility

The need for flexibility in systems is due to several reasons including:

- Extending of operational lifetimes.
- Rapid changes in technology, budgets, national objectives, threats.
- The desire to minimize the number of different components in the inventory to simplify the logistics and maintenance aspects of Sustainment.

[43] See Chapter 13.

11.3 Requirements for flexibility

Consider the requirements for flexibility Flexible systems are those that can be used for more than one purpose or mission, and especially purposes that are developed after the systems have been deployed. While flexibility is desirable in systems, for any specific purpose, a system designed for that purpose would usually be more efficient than a multipurpose system. This facet can readily be observed in the ubiquitous Swiss army knife. While it can do many things, each function can be performed more easily with a tool specifically designed for the function. Flexible systems have a high probability of being used in scenarios that were unknown at the time the systems were designed. Flexible systems tend to be used in a "design to inventory" situation in which a problem scenario is met using equipment to hand. Thus the more flexible the equipment can be made, the more scenarios it can be used for. The principle of flexibility applies to refined or derived requirements in the design phase of a system as well as in the traditional system requirements. In order to be able to write requirements for flexibility, the attributes of flexibility need to be considered.

11.4 Attributes of flexibility

Attributes of flexibility determine the degree of reuse or compatibility with other systems. One major attribute of flexibility is a "standard" for interoperability and requirements for flexible systems must specify the use of appropriate standards. Consider the following examples.

- Table driven computer software
- Rack mounting hardware
- Standard Connectors
- Data format standards
- Size and shape of ordnance
- Bus connectors in computer based systems

11.4.1 Table driven software

Consider software used to display telemetry information on the screen of an operator's terminal. The software could be specifically programmed for displaying different types of information (text, numbers, etc.) using a number of different subroutines or a single generic table driven (parameter) module could be used. For example, if a display routing was developed that accepted:

- X coordinate;
- Y coordinate;
- Data to be displayed;
- Type of data (numeric, text, etc.);

- Colour of data;
- Format and decimal point information if numeric;
- Lower limit value;
- Lower limit colour;
- Upper limit value;
- Upper limit colour;

The routine then becomes reusable in other applications and programs where data has to be displayed at various locations on a screen and the colour of the information changed according to different criteria. The cost of developing and testing custom data display software modules is then avoided for future programs.

11.4.2 Rack mounting hardware

When hardware is made to fit in 19-inch racks, any facility equipped with the racks can be used for multiple purposes.

11.4.3 Standard connectors

The use of standard connectors allows many types of equipment made by a variety of manufacturers to be used interchangeably. The ubiquitous TV antenna connector, telephone and Ethernet connectors are typical examples.

11.4.4 Data format standards

These allow for the interchange of information between equipment.

11.4.5 Size and shape of ordnance

Bullets of a specific calibre made by a variety of vendors may be fired from a variety of weapons also made by a variety of vendors.

11.4.6 Bus connectors in computer based systems

A bus allows the user to add capability not present in the system. Thus the bus in a personal computer (PC) allows for the addition of capability that may not exist when the unit was manufactured. The evolution of the PC can be told by the story of the migration of functions from daughter boards to the motherboards once the functions became “standard”.

11.5 Capability drives requirements

In the traditional acquisition process, a system that met the requirements was implemented within a set of constraints. However, the set of constraints did not include that of architecture, so the designers were free to

build their system around "any" architecture. However in an age of "Design to Inventory" the designers now have to limit their implementation to something that exists within the relevant architectures. Thus the system engineer now has to identify non-existent capability within the architectural framework. This is done by an iterative process in the following manner.

1. Develop a set of operational scenarios or concepts for the system and its adjacent systems in the context shown in Figure 11-1.
2. For each item (system) in the inventory, identify in which of the scenarios it can be used.
3. When done, draw a table similar to Table 11-1. Use an "X" to signify that the inventory item has capability that allows it to be used in the scenario.

Inventory Item	Scenario				
	1	2	3	4	5
A		X	X	X	
B	X		X	X	X
C		X		X	X
D					X
E	X		X		X

Table 11-1 Capability of inventory items in various scenarios

4. Examine the rows and columns in the table. Inventory items (rows) that have X's marked in several columns are flexible items. Any row that is a subset of another row contains an item that is less flexible than the other row and is a candidate for phasing out unless there are specific reasons for not doing so (i.e. much lower in cost or has additional capability not used in any of the scenarios).
5. Examine the capability of the items and use the total capability to postulate additional scenarios. These scenarios to be based on evolution of technology, changes to national objectives and threats, etc.
6. Add columns for the additional scenarios to the table.
7. Insert an "X" in the appropriate place for the items that can be used in the new scenarios.
8. Determine missing capability.
9. Start acquisition process for systems that would contain the missing capability (a column without an "X").

10. Go back to step 6.

This process is that of an evolutionary acquisition paradigm in which the system being acquired is considered in the context of its adjacent systems.

Capability is word for which there is no consensus on its definition, however there is agreement that it does consist of a mixture of equipment, personnel and the preparedness and ability of the personnel to perform. Mueller incorporates this mixture in the following definition:

"Defence Capability can be defined as the power to achieve a desired operational effect in a nominated environment within a specified period of time and to sustain that effect for a designated period" (Mueller, 2001).

Mueller states that the essential theme is that of taking the whole system view during every phase of the lifecycle to optimise the system, rather than trying to optimise individual parts and then integrate inputs." Thus in the development of Defence Capability, there may need to be trade-offs between parts of the mixture, such as the complexity of operation of equipment and the amount of training needed[44]. Each of these parts is defined as a system in itself and at any given time, the individual elements are in a mix of being acquired, being brought into service, being operated and maintained, and being phased out. This is one definition of a System of Systems[45] namely "*a system made up of elements that are not acquired or designed as a single system but are acquired over time and are in continuous evolution*" (Allison and Cook, 1998). Permanent examples of this definition of a System of Systems (Allison and Cook, 1998) are:

- Airlines.
- National Defence forces.

Temporary, ephemeral or virtual examples of such a System of Systems (Allison and Cook, 1998) are

- Multi-national peace keeping forces.
- Project teams.

However, Hichins' five layers of systems engineering[46] define the context for, and the type of, system in each layer, rather than use the term Systems of Systems for the upper layers. Each system that makes up Defence

[44] Think of Defence Capability as a system experiencing the iterative multi-phased acquisition cycle depicted in the process for the engineering of complex systems described in Chapter 13.
[45] For a discussion on the myth of Systems of Systems see Section 26.6.
[46] See Section 12.1.2.

Capability operates within the context or framework of its adjacent systems. The physical context perspective is as shown in Figure 11-1 and the temporal context perspective is as shown in Figure 11-2.

11.6 Just in time requirements

Along with the evolutionary approach, some requirements must also be finalized using a just-in-time (JIT) approach manner (Davies, 1998)[47]. The JIT approach was developed by the Ford Motor Car Company to reduce the amount of capital tied up in raw materials (Ford and Crowther, 1922) page 143). The use of JIT requirements serves a different purpose. There are many scenarios in which the functionality required can be met by today's technology and locking in a requirement to use the technology will lead to poor performance in the future. For example, systems have been delivered within the last few years with built-in 386 processors. The requirements were written at a time when the 386 was the latest and greatest, but by the time of delivery technology had long since passed the 386. Now there may be good reasons for requiring specific instances of technology, however in most cases delaying the decision works in the interests of the customer. Delaying decisions is risky and such related decisions should be flagged in the project management information system to minimize risks. Simple examples of the negative aspect of using fast processors were:

- Software written in Borland's Turbo Pascal will not run on modern faster processors due to delay loops embedded in the compiler failing to provide enough delay in the serial port interface.
- The IBM PC compatible ATs were released with CPU speed switches. The purpose of the switch was to slow down the clock rate so that games that ran on the original XT and used software timing loops would still be useable on the AT.

In general the process is to examine the situation to determine the level of risk. If the requirement can be met by a number of solutions, then the decision as to which solution to implement may be deferred. As an example, if the requirement is to provide a long-distance communications link, then it may be met using a variety of technologies such as communications satellite, High Frequency radio, microwave link, optical fibre. If several of the technologies will meet the requirement there is little risk in delaying the decision and the capability provided by the technologies in existence at the time the decision will be made will probably provide the customer with a more flexible system (extra capability).

[47] And discussed in Chapter 15.

11.7 The backcasting approach

Another way to identify and develop flexible systems is to work back from the solution. Traditionally we are taught to work forwards, in that several (implementation-free) designs are proposed in response to the requirements and the optimal solution is then chosen. An alternative is to work back from the solution (Gouillart and Kelly, 1995) page 49). The process is:

- Visualize the system in operation (try to focus on what the system does rather than how it does it). This is the operations concept.
- Develop the system requirements.
- Identify the requirements that can be implemented in a JIT manner.
- Identify the "don't care" requirements.
- Identify the "don't want" requirements.
- Pick a design that implements the "want" and "don't want" requirements and tends not to inhibit the "don't cares".
- Design the system making JIT decisions.

Implementing a backcasting approach requires the three factors listed below. Since organizations with low CMM levels probably do not have a suitable change control process, backcasting may only be appropriate for higher-level CMM organizations. The three factors are:

- **A contextual attitude** –currently thought of as a System of Systems attitude. External factors that could affect the system must be identified and monitored and changes to the project made as appropriate.
- **An effective change control process** - The change control board must make sure that decisions are made in a timely manner.
- **The ability not to fixate on a single solution** – something may happen during the implementation phase which may require the approach to be cancelled or significantly modified (e.g. technology break-through or change to mission/business directions). Should this situation arise, the CCB must act appropriately and not ignore the change in the context of the project.

11.8 Examples of flexible and non-flexible systems

Consider the following sample of flexible and non-flexible systems from the macro to the micro level.

- The LuZ SEGS-1 Project.
- NASA's Space Transport System.
- The Division Air Defense Sergeant York gun.
- The Standard Central Air Data Computer (SCADC).
- The RCA Cosmac Microprocessor.
- The F-35 Joint Strike Fighter

- The USAF's F-1000 engine

11.8.1 The Luz SEGS-1 Project

In the mid-1980's the LuZ Group, a start-up joint Israel-American venture were developing the world's first commercial solar electrical power generating system (SEGS-1) (Kasser, 1984). As the first of its kind, SEGS-1 initially only existed then as a vague concept. The station was to be installed in the Mojave Desert in California and the Research and Development was to be in Jerusalem. SEGS 1 was intended to generate electrical power from the sun by focussing the sun's rays on about 600 parabolic mirror trough reflector collectors each about 40 meters long. The operation of each parabolic trough reflector was monitored and controlled by a microprocessor based local controller (LOC). Each LOC controlled a motor that positioned the parabola, and received information about the angle of elevation and the temperature of the oil in the pipe positioned at the focus of the trough. Oil was pumped through the piping, and as long as the LOC kept the reflector pointed at the sun within an accuracy of ±0.2 degrees, the oil was heated. The hot oil was pumped thorough a heat exchanger to generate steam. The steam drove a turbine that generated up to 15 Megawatts of electrical power. Figure 11-3 shows a partial depiction of the system. Although it was a complicated system, it still had a conversion efficiency of about 40%, greater than any alternative method of harnessing solar energy at the time. The situation was very uncertain (flexible), namely:

- Control would be single axis (elevation) only.
- Power generation efficiency dropped rapidly if the mirrors were not pointing at the sun. Actually the mirrors would radiate the heat instead of absorbing it under off-pointing conditions.
- There were wide tolerances on the North-South alignment of collectors.
- The electromagnetic (E-M) interference environment at the site was unknown.
- The sun sensors were mounted on the mirrors.
- There was an experimental heliostat based solar power generating plant somewhere close to the site, and it was felt that the LOC's sun sensors could lock onto the heliostat tower under some unspecified conditions.
- There were no vibration specifications for the mirrors as a result of movement or for any other cause.
- At that time, Jerusalem was at the end of a long delay in purchasing parts due to geographical distance. This meant that purchase orders for prototyping parts had to be placed before the designs were completed.
- The engineers and technicians spoke various combinations of Russian, English, French, Romanian, and Hebrew, because most were immigrants and there were times when there was no common language in a meeting.

Figure 11-3 Part of the LuZ SEGS-1 system

- There were no production facilities for electronic circuits.
- There were few if any quality control concepts.
- There were few if any written specifications or procedures in the control and electronics department
- There were some barely working prototypes.

The control system was designed by optimising the system as a whole, using an object-oriented approach and allowing for uncertainty. Thus:

- The central computer predicted the approximate position of the sun and commanded the LOCs to deploy to that appropriate position.
- The LOCs were designed as "state machines" (the states being - rest, deploying, acquiring the sun, tracking, and stowing).
- Each LOC was able to sense (acquire) the sun using the sun sensor mounted on the mirror.
- Each LOC was able to follow the movement of the sun using a lead-lag algorithm to move along with the sun to "flywheel" during short periods of cloud cover. Transferring the pointing functionality from the central computer to the individual LOCs meant that the planned mini-computer could be replaced by a $2,000 microcomputer[48]. This approach avoided at least $US 300,000 in hardware and software costs for each of three central control stations (CCS), namely about $900,000. In addition if the

[48] This was still in the days of the 8-bit microprocessor.

control link failed the mirrors would continue to point at and track the sun.

- The sensors used to detect the elevation angle of the mirrors were absolute position indicators instead of relative indicators based on a revolution count. This design approach allowed the LOCs to recover quickly in the event of power failures and spikes on the power line in the unknown E-M environment. An alternative turn counting design would require that the mirrors return to the Stowed position to reset the counters after a power failure losing heat during the reset sequence.
- The communications approach between the central computer and the LOCs was based on a sequential polling approach at relatively slow speeds using shielded twisted pair connectors and human readable ASCII text messages.
- This approach allowed for servicing using palm held ASCII terminals that eliminated the need to develop special test equipment. Low data rate speeds were valid because the movement of the mirrors was very slow, as was the rate of heating of the oil.

However, there while the concept of flexibility worked well in the control and electronics area, it was not employed in other departments. The sun sensor provided an example of what can go wrong. The sun sensor used a lens to focus the sun onto a pair of photo diodes with a separating spacer as shown in Figure 11-4. During the assembly process, the diodes were glued to a base plate with transparent glue. The physics department who were building the sun sensors did not place a requirement that there be no glue on the side of the diode illuminated by the sun. After all, the glue was transparent. A year or so later, they found that the glue slowly became opaque when subjected daily to the very high temperature at the focal point of the lens. This phenomenon resulted in the need to replace all the sun sensors. From a manufacturing perspective, there was little difference in mounting the diodes if the glue could or could not be allowed to cover the face of the diode, just a matter of care and a few extra minutes of time. Nobody asked about possible changes to the characteristics of the glue over long periods of time under high temperature. If the requirement had been placed on the process, not to allow glue on the face of the diode, the characteristics of the glue under the high temperature conditions would not have mattered and the expensive sun-sensor replacements would have been avoided (Kasser, 1995). This is an example of introducing an unnecessary failure mode by not utilizing the “don’t cares”. Thus the lesson learned is that if it doesn’t make any difference don’t do it.

Chapter 22 discusses some of the problem-solving experiences in portions of the LuZ system SDLC.

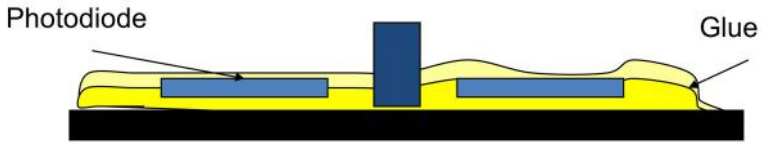


Figure 11-4 The Luz sun sensor

11.8.2 NASA's Space Transportation System

The Space Transportation System commonly known, as the Space Shuttle is an example of an approach to designing a flexible system based on the assumption that "one size fits all". Concerned about the cost of expendable vehicles in an era when space flights were anticipated to become commonplace, NASA intended to reduce costs by designing a single reusable vehicle. The result was an expensive vehicle and not an entirely reusable one since expendable parts (booster and fuel tanks) are employed.

Experience has shown that one size does not necessarily fit all and a more flexible solution may be expendable cargo carriers and a small reusable manned vehicle. The manned vehicle was to be a single stage to orbit type of vehicle with the support requirements of a commercial passenger jet aircraft. However such a solution for the shuttle replacement may still not be politically correct since NASA has a large investment in manned mission support. In a sense, since NASA does launch both the shuttle and expendable vehicles, they have implemented a mixed solution but at significantly greater cost than a system designed for the mixed scenario. Once on-orbit-docking capability had become routine, the mixed launcher solution became very viable from a technical perspective. Yet the political impact of the reduction in support staff is a pertinent issue that seemingly remains. The shuttle fleet is aging and no replacement program has been announced even with the commitment to support the International Space Station.

11.8.3 The Division Air Defense Sergeant York Gun

The Division Air Defense (DIVAD) gun program was initiated by the US Army's need for a replacement for the aging M163 20mm Vulcan Anti-Aircraft gun and M48 Chaparral missile systems (Pike, 1999). The new self-propelled anti-aircraft gun system was to be based on the M48A5 tank chassis, using as much off-the-shelf equipment as possible. After examining two designs, the Army selected Ford Aerospace and Communications Corporation as the contractor in May 1981 and the gun was designated as the M247 Sergeant York. Ford Aerospace's design was based on the reuse of two additional existing systems (capability) namely

- **The weapons** - twin 40mm L/70 Bofors Guns.

- **The radar** - a modified version of the Westinghouse APG-66 system used in the F-16 Fighting Falcon.

The program suffered from problems, and by the delivery of the first production vehicles in 1983 many problems remained, including:

- The radar's inability to track low flying targets due to excessive ground clutter. The radar could not distinguish between a hovering helicopter and a clump of trees.
- When tracking highflying targets, the radar return from the gun barrel tips confused the fire control system. Turret traverse was also too slow to track a fast crossing target.
- The electronic counter-measures (ECM) suite could be defeated by only minor jamming.
- The use of the 30-year-old M48 chassis design meant the vehicle had trouble keeping pace with the newer M1 Abrams and M2/3 Bradley's, the very vehicles it was designed to protect.

These problems proved insurmountable, and in December 1986 after about 50 vehicles had been produced the entire program was terminated.

11.8.4 The Standard Central Air Data Computer

The Standard Central Air Data Computer (SCADC) project (Howard, 2001) was one of about a dozen standardization programs initiated in the late 1970's by the US DOD in the desire to obtain substantial reductions in equipment life-cycle costs through the wide use of digital common modules in aircraft. It was thought that SCADC, because of the complexity and accuracy requirements of air data computation, would be a difficult concept to bring to fruition. In addition, the SCADC program required the delivery of up to 150 units per month shared between two winning suppliers, in a continuously competitive leader-follower arrangement.

Two of the three largest US suppliers of airborne air-data systems, Honeywell and Sperry declared the concept impossible and declined to bid, despite the potential of $500 million of business. GEC Avionics in the UK were interested in the business and designed and built a SCADC in a replacement form fit-function for the then existing analogue components. The system was a modular core set of standard Air Data Computer modules made extendable by the use of the 1553 data bus.

The ability to replace "old for new" in around 30 minutes on thousands of the older inventory aircraft, raising the Mean Time Between Failure (MTBF) rates from about 100 hours to greater than the aircraft operational life-times and at the same time equipping them for plug-in new attack systems (via the 1553 data bus) was a significant technical innovation. However, it also had the effect of putting many logistics people out of work overnight. When the first prototypes were demonstrated, a huge effort was

launched in Washington by the Logistics fraternity to have the project cancelled. This was supported by much of the US industry who could see an outcome that depleted a large portion of their diverse business with the danger of much of it going overseas. Although the SCADC production programs continued, the implementation in service was delayed for up to two years. Another casualty was that all the other standardization programs fell by the wayside.

By 1998, 6000 units in various configurations had been sold including a version now modified into a digital flight control system adopted by the US Navy for the F14. It was the most widely used digital system and most reliable in all aircraft in the Gulf War. The program was arguably the most successful of any airborne equipment supply program in the history of world aerospace, and 100% of the production units came from the UK source, the leader-follower concept being abandoned. It was estimated that by the time GEC Avionics received the third or fourth order for production units, the direct Defence savings exceeded $US 500 million.

11.8.5 The RCA Cosmac Microprocessor

The Cosmac was the first 8 bit CMOS microprocessor in the days when all the rest were NMOS (mid 1970's). It was ideal for environments in which the low power and other characteristics of CMOS were desirable. However, it was less than a success. The hardware circuitry was simple to use, however, the software architecture was extremely flexible. It was so flexible that any register could be programmed to serve any function. Most people ended up using the device in fixed configurations and did not take advantage of the flexibility provided by RCA.

11.8.6 The F-35 Joint Strike Fighter

The Joint Strike Fighter (JSF) is a multi-role fighter optimised for the air-to-ground role, designed to affordably meet the needs of the USAF, US Navy (USN), US Marine Corps (USMC) and allies, with improved survivability, precision engagement capability, the mobility necessary for future joint operations, and the reduced lifecycle costs associated with tomorrow's fiscal environment. The JSF will benefit from many of the same technologies developed for F-22 and will capitalize on commonality and modularity to maximize affordability (Pike, 2000).

The JSF program will demonstrate two competing weapon system concepts for a tri-service family of aircraft to affordably meet these service needs. However the needs are different as summarized in Table 11-2, namely:

Service	Variant	Cost FY94 US$
U.S. Air Force	Conventional Take-off and Landing (CTOL)	$28M
U.S. Marine Corps	Short Take-off and Vertical Landing (STOVL)	$35M
Royal Navy (UK)		
U.S. Navy	Carrier-based (CV)	$38M

Table 11-2 JSF variations

- The USAF wants a multi-role aircraft (primarily air-to-ground) to replace the F-16 and A-10 and to complement the F-22. The USAF JSF variant poses the smallest relative engineering challenge. The aircraft has no hover criteria to satisfy, and the characteristics and handling qualities associated with carrier operations do not come into play. As the biggest customer for the JSF, the service will not accept a multi-role F-16 fighter replacement that doesn't significantly improve on the original.
- The USN wants a multi-role, stealthy strike fighter to complement F/A-18E/F. Carrier operations account for most of the differences between the Navy version and the other JSF variants. The USN version of the aircraft has:
    - Larger wing and tail control surfaces to better manage low-speed approaches.
    - A strengthened internal structure to handle the loads associated with catapult launches and arrested landings.
    - A carrier-suitable tail hook.
    - Landing gear with a longer stroke and higher load capacity.
    - Almost twice the range of an F-18C on internal fuel.
    - A design that is also optimised for survivability.
- The USMC wants a multi-role Short Take-Off & Vertical Landing (STOVL) strike fighter to replace the AV-8B and F/A-18A/C/D. The Marine variant distinguishes itself from the other variants with its short take-off/vertical landing capability.
- The United Kingdom's Royal Navy is looking for a STOVL (supersonic) aircraft to replace the Sea Harrier. The Royal Navy's JSF will be very similar to the USMC variant.

The JSF concept is to build these three highly common variants on the same production line using flexible manufacturing technology. Cost benefits were expected to result from using a flexible manufacturing approach and

common subsystems to gain economies of scale. Cost commonality was projected in the range of 70-90 percent; parts commonality will be lower, but emphasis is on commonality in the higher-priced parts. The decision to reuse many of the technologies developed for the F-22 on the JSF was made before the problem of obsolescence was widely noticed. Thus the lifetime of many of the F-22 components will be extended by their incorporation into the JSF so something needs to be done about reducing their cost of obsolescence[49].

11.8.7 The USAF's F-1000 engine

Some of the reliability and performance problems that plagued the USAF's F-1000 engine stemmed from the engine's remarkable power and resilience. Whereas the engine had been designed principally for speed, pilots of the new F-15, the first aircraft powered by the F-1000 engine, found operational advantage in rapidly changing speed; thus submitting the engine to "thermal cycles" not envisioned as the engine was designed and tested. This was one of several reasons the engine underwent nearly a decade's worth of maturational development before reaching its final potential (McNaugher, 2000).

11.9 Lesson Learned from these systems

Various lessons can be learnt from these systems including:

- Programs do not fail because the requirements change, programs fail due to poor change management.
- There are risks when using capability in operational ways for which it was not designed.
- Political considerations outweigh technical factors.
- The context in which the system is being acquired must be taken into account.
- Too much flexibility can be worse than no flexibility.

11.9.1 Programs do not fail because the requirements change, programs fail due to poor change management

The tasks, products, and processes for managing change exist and have done so for over 80 years (Farnham, 1920) and the US Military specifications cover the ground in more than enough detail. Programs fail because of poor requirements engineering management[50] and the failure to re-evaluate re-

[49] Note Stevens stated the then current estimate of the parts availability lifetime for the F-22 was approximately 2 to 5 years (Stevens, 2003).
[50] The B paradigm is inherently flawed (Chapter 28).

quirements in the context of:

- Changes in needs.
- Changes in technology.
- Changes in paradigm. The effect of air power on battleships was noted in World War II. Today a paradigm shift is taking place due to the development of low cost guided ordnance. Their effect on the JSF and other expensive aircraft as well as surface ships has yet to be fully determined.

11.9.2 There are risks when using capability in ways for which it was not designed

These range from the USAF's F-1000 engine to the DIVAD. Sometimes the excess capability embodied in flexibility leads to problems when equipment is used out of context. Warriors use new equipment in ways that were not envisioned by designers or tested in formal testing. Although this is a tribute to the work of developers whose systems invite novel uses, it nonetheless can produce embarrassing and costly technical problems as in the case of the F-1000 engine (McNaugher, 2000).

In the DIVAD the integration of three systems produced an array of technical and operational test difficulties (undesirable emergent properties) that ultimately led to its cancellation. Some of these difficulties might have been prevented had the situation been different. For example, the pilots must have known about the problems with the flight radar at low altitudes, yet the information was not passed to Ford Aerospace or the selection board. There is thus a need for domain expertise when attempting to use capability outside its original environment.

In addition, four successive generations of upgraded US forward area air defence systems - from Mauler to Roland to Sgt. York to the Air-Defense Anti-Tank System (ADATS) - were all cancelled, at a total cost of more than $6.7 Billion over a period of 30 years (Pike, 1999). What does this mean in an evolutionary acquisition paradigm in an era of extending system's operational lifespan?

11.9.3 Political considerations outweigh technical factors

This lesson reinforces the findings in Chapter 5. The optimal technical solution may be resisted for political reasons, as in the SCADC and space shuttle programs.

11.9.4 The context and environment must be taken into account

The advances in the capability of the vehicles the DIVAD was to support would have been observed and affected the program had the project considered changes in the context and environment in which the DIVAD was to be deployed. Not only must the context and environment be taken into ac-

count, the project must focus on the real needs not the solution unlike in the DIVAD program. Take manned aircraft as another example. In World War I they provided ability to see over the horizon and bring back information about enemy dispositions. Then each side added guns to their aircraft to stop the enemy from doing the same. This led to squadrons of fighter aircraft fighting each other but in reality doing little to advance the war effort. Guns also allowed ground support, which led to ordinance delivery on the trenches and behind the lines. These situations led to specialized types of aircraft, complex on-board computers, and latterly to the JSF. Yet the question that really needs to be asked is - what are today's real mission requirements and how can they best be met (with or without manned aircraft)?

11.9.5 Too much flexibility can be worse than no flexibility

In this situation the frame of reference is lost and the flexibility ends up providing too many choices. This is probably why the RCA Cosmac microprocessor failed to be adopted widely.

11.10 Conclusions

The conclusions from this study are that:

- Writing requirements for flexible systems is not easy and is pointless unless effective intelligent change management tools are developed and used.
- History repeats itself. The JSF is doomed to cost and schedule overruns due to the lack of intelligent change management tools, as well as the political considerations involved.
- The JSF may also be doomed in threat environments due to failure to recognize a paradigm shift due to the introduction of guided ordnance.
- One size does not fit all, several may be required. This is an important lesson which we repeatedly fail to learn from experience[51].
- The amount of flexibility or excess capability incorporated into a system needs very careful consideration.

[51] ADA failed to become the DOD computer programming language, and UML will fail to become the object-oriented world's single descriptive language for similar reasons.

2001

12 A framework for a systems engineering body of knowledge

The demand for systems engineers and taught postgraduate degrees in systems engineering (by coursework) is growing around the world demand (Fabrycky, 2003). However, meeting that demand is not a simple affair[52]. There is a scarcity of qualified personnel who truly understand the nature of systems engineering and can teach systems engineering subjects in academia. This Chapter:

- Looks at systems engineering from the academic perspective shows that systems engineering has no recognized body of knowledge[53] which explains why systems engineers have difficulty agreeing on exactly what systems engineering should be and academia while teaching it has difficulty in deciding if it is an undergraduate or a post graduate degree and what subjects should be included in the course[54.]
- Summarizes a number of models of systems engineering and proposes a framework for a SEBoK based on a combination of two of the models.
- Proposes a road map for the development of the SEBoK.
- Closes with some perspectives on systems engineering that become visible when system engineering is viewed within the proposed framework.

At present "systems engineering":

---

[52] Although this Chapter has a focus on the academic teaching for systems engineering, it would be appropriate to ask whether this is the best approach. In particular one could ask whether this discipline is better developed via in-house training rather than in a university. For example, it is worth noting that there is some evidence (e.g. McCornick in R&D Management #2, 1995) that the Japanese value internal more than external training.

[53] By the systems engineers themselves.

[54] Actually this is may be a little broader since one may also need to identify the prerequisites for starting to understand system engineering.

- Covers a broad spectrum of activities from soft systems and organizations to hard computer based systems.
- Is a vague term with many different interpretations. Table 12-1 contains a number of definitions of systems engineering. From one reading these definitions, it may appear that the state of the art of systems engineering appears to be comparable to the state of electrical engineering before the advent of Ohm's law (Chapter 2).
- Has a number of process standards and CMMs which are focused on both the entire set and various subsets of activities involved in the development of computer-based systems.
- Has no standard for competence (Chapter 7)[55].

Table 12-1 A selection of definitions of systems engineering as published in chronological order

The combination of advanced chemical engineering science with the tool of electronic computers and the viewpoint of considering the process as an entity (Williams, 1961).
Systems engineering considers the content of the reservoir of new knowledge, then plans and participates in the action of projects and whole programs of projects leading to applications. It considers the needs of its customers and determines how these can best be met in the light of all knowledge both old and new. Thus systems engineering operates in the space between research and business, and assumes the attitudes of both. For those projects which it finds most worthwhile for development, it formulates the operational, performance and economic objectives, and the broad technical plan to be followed (Hall, 1962) page 4).
The science of designing complex systems in their totality to ensure that the component sub-systems making up the system are designed, fitted together, checked and operated in the most efficient way (Jenkins, 1969).
Systems engineering covers the comprehensive aspects of engineering practice, and the application of the modern rational approach to the formulation and solution of technical problems (Au and Stelson, 1969) page 1)
The application of scientific and engineering efforts to transform an

[55] See updated research findings in Chapter 24.

operational need into a description of system performance parameters and a system configuration through the use of an iterative process of definition, synthesis, analysis, design, test, and evaluation; integrate related technical parameters and Ensures compatibility of all physical, functional, and program interfaces in a manner that optimises the total system definition and design; integrate reliability, maintainability, safety, survivability, human engineering, and other such factors into the total engineering effort to meet cost, schedule, supportability, and technical performance objectives.(MIL-STD-499A, 1974).
The transforming of an operational need into a description of system performance parameters and a system configuration. (FM_770-78, 1979)
A hybrid methodology that combines policy analysis, design and management. It aims to ensure that a complex man-made system, selected from the range of options on offer, is the one most likely to satisfy the owner's objectives in the context of long-term future operational or market environments (M'Pherson, 1986) pages 330-331).
An iterative process of top-down synthesis, development, and operation of a real-world system that satisfies, in a near-optimal manner, the full range of requirements for the system (Eisner, 1988) page 17).
The management function which controls the total system development effort for the purpose of achieving an optimum balance of all system elements. It is a process which transforms an operational need into a description of system parameters and integrates those parameters to optimise the overall system effectiveness (DSMC, 1996) pages 1-2).
A robust approach to the design and creation of systems to accomplish desired ends (Chamberlain and Shishko, 1991) page 23).
An interdisciplinary approach to evolve and verify an integrated and lifecycle balanced set of system product and process solutions that satisfy customer needs. Systems engineering: • encompasses the scientific and engineering efforts related to the development, manufacturing, verification, deployment, operations, support, and disposal of system products and processes, • develops needed user training equipments, procedures, and data,

• establishes and maintains configuration management of the system, • develops work breakdown structures and statements of work, and • provides information for management decision making (MIL-STD-499B, 1992).
A management technology (Sage, 1992) page 1).
The design, production, and maintenance of trustworthy systems within cost and time constraints (Sage, 1992) page 10).
The intellectual, academic and professional discipline the principal concern of which is the responsibility to ensure that all requirements for a bioware/hardware/software system are satisfied throughout the life of the system (Wymore, 1993) page 5)
Integrates all the disciplines and specialty groups into a team effort forming a structured development process that proceeds from concept to production to operation. Systems engineering considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs (Sage, 1992).
A discipline created to compensate for the lack of strategic technical knowledge and experience by middle and project managers in organizations functioning according to Taylor's "Principles of Scientific Management" (Chapter 2).
Comprises systems analysis, systems integration and human factors including human-computer interaction.(Anderson and Dibb, 1996)
The activity of specifying, designing, implementing, validating, installing and maintaining systems as a whole (Sommerville, 1998).
A complex collection of interactive units and subsystems within a single product, jointly performing a wide range of independent functions to meet a specific operational mission or need (Shenhar and Bonen, 1997).
A set of activities which control the overall design, development, implementation and integration of a complex set of interacting components or systems to meet the needs of all the users (DERA, 1997).
An interdisciplinary approach and means to enable the realisation of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting

requirements, then proceeding with design synthesis and system validation while considering the complete problem. (INCOSE, 2000).
The design and analysis process which decomposes an application into software and hardware (Brodie, 2001) page 249)
The process that identifies the technical characteristics and operating rules of that system that best achieves the objectives in question (Westerman, 2001) page 6).
The art and science of creating systems (Hitchins, 2003)
Deals with the planning, development, and administration of complex systems, particularly of computing systems (Endres and Rombach, 2003) page 1).
Provides a framework, within which complex systems can be adequately defined, analysed, specified, manufactured, operated, and supported (Faulconbridge and Ryan, 2003).
Guides the engineering of complex systems (Kossiakoff and Sweet, 2003)

As such,

- Many systems engineers cannot clearly articulate the functions and benefits of systems engineering (Chapter 10).
- It has been extremely difficult to establish a SEBoK for the diverse activities that are known by the term "systems engineering."

As of 2003, there were over 100 postgraduate programs in the field catering to an ever-growing demand (Fabrycky, 2003). Even though there is no widely recognized SEBoK, several starts have been made in assembling a SEBoK including:

- A curriculum which has been presented for teaching a few subjects in the context of an undergraduate degree (Faulconbridge and Ryan, 1999).
- A reference curriculum under development in 2006 by the Education and Research Technical Committee (IERT) of INCOSE to inform teaching at universities and training needs within a workplace-based employee competency framework.

12.1 Potential Frameworks

Without a framework in which to place the collection of knowledge, the information may be incomplete. Several models that have the potential to act

as a contextual framework for a taxonomy of the SEBoK have been published and the following models are briefly discussed in this Chapter:

- Allison and Cook's systems hierarchy.
- Hitchins' Five-layer Model.
- Sage's Three Overlapping facets Model.
- Badaway's Master of Technology.
- Kasser's People Process Product Time (PPPT) enterprise framework.

12.1.1 Allison and Cook's systems hierarchy

Allison and Cook proposed that military systems thinking and practice needs to be extended to encompass a systems hierarchy which includes systems and System of Systems) etc. (Allison and Cook, 1998). They proposed that the focus is no longer only on the acquisition and use of individual systems, but now there is a need to consider the "forest" of systems and how they can be integrated in a variety of ways to satisfy different military purposes. This extension of systems engineering presages the creation of new methods and tool sets to support the new activities. In moving to address total capabilities, systems engineers need to move away from acquisition of individual systems, to the acquisition of super systems that are not obtained through a single acquisition exercise, not necessarily under the direction of a single authority. They proposed an approach based on architectures in which military enterprise architecture[56] would be developed that is in reality a dual architecture:

- **The Preparedness Architecture** - describes the tasks and Defence elements needed to develop, train, and prepare the Force, and the relationships, interactions, and information flows between these elements.
- **The Joint Operations (or War fighting) Architecture** - describes those tasks, operational elements, and information flows that support actual operations (war fighting).

12.1.2 Hitchins' Five-layer Model

Hitchins proposed the following five-layer model for systems engineering (Hitchins, 2000):

- **Layer 5** – Socio-economic, the stuff of regulation and government control.

[56] The military enterprise architecture view is defined as "a description of the tasks and activities, military elements, and information flows required to accomplish or support a military function or operation".

- **Layer 4** - Industrial systems engineering or engineering of complete supply chains/circles.
- **Layer 3** - Business systems engineering - many businesses make an industry. At this layer, systems engineering seeks to optimise performance somewhat independent of other businesses.
- **Layer 2** - Project or System layer. Many projects make a Business. Western engineer-managers operate at this layer, principally making complex artefacts.
- **Layer 1** - Product layer. Many products make a system. The tangible artefact layer. Many engineers and their institutions consider this to be the only "real" systems engineering.

Hitchins states that the five layers form a "nesting" model, i.e. many products make a project, many projects make a business, many businesses make an industry and many industries make a socio-economic system. Hitchins adds that clearly, these statements are only approximate since-

- A socio-economic system has more in it than just industries.
- A business has more in it than just projects, and so on.
- Actual organizations may divide the work in different ways resulting in either sub-layers, or different logical break points.

12.1.3 Sage's three overlapping facets model

Sage suggests that system engineering needs to be dealt with as three overlapping facets … namely of structure, function and purpose which overlap (Sage, 1992). Sage provides a definition for each and then divides each as follows:

- **Purpose** - Systems Management and consists of enterprise, process reengineering, process maturity, organizational environment, organisational culture, strategic costs and effectiveness metrics, benchmarking and strategic quality (TQM).
- **Structure** - Systems Methodology and consists of lifecycles, concurrent engineering, structural effectiveness metrics, decision assessment, structural economic analysis, cognitive ergonomics, configuration control and quality assurance.
- **Function** - systems engineering Methods and Tools and consists of performance metrics, control and communications theory, requirements engineering, functional economic analysis, programming languages, simulation and modelling, operations research and quality control and statistics.

If the actual core discipline of "programming" at the lowest layer was replaced by "core specialist knowledge" this structure could be used for a SEBoK. However, each of the three facets would still be complicated.

12.1.4 Badaway's Master of Technology

Badaway argues that the need is for "engineering management" training, and describes a possible postgraduate Master's degree that is a hybrid mix of a Master of Business Administration (MBA) and a Master of Engineering that he calls a Master of Technology (MOT) (Badaway, 1995). However, it has a high level of overlap with systems engineering. The starting place is that the National Research Council defines MOT as linking "*engineering, science and management disciplines to address the issues involved in the planning, development and implementation of technological capabilities to shape and accomplish the strategic and operational objectives of an organisation*". In particular "shaping" is very important since most systems engineering starts with the idea of filling a pre-defined requirement, but in reality the technology often produces new capability that meets what no one had envisaged as a requirement when the project began. The overall framework he proposes is to meet the following eight primary needs.

- How to integrate technology into the overall strategic objectives of the firm.
- How to get into and out of technologies faster and more efficiently.
- How to assess/evaluate technology more efficiently.
- How to accomplish technology transfer.
- How to reduce new product development time
- How to manage large, complex and interdisciplinary or inter-organizational projects/systems
- How to manage the organizations internal use of technology.
- How to leverage the effectiveness of technical professionals.

12.1.5 Kasser's people process product time (PPPT) enterprise framework

Robert Frosh, when he was Assistant Secretary to the United States (of America) Navy wrote: "*Systems, even very large systems, are not developed by the tools of systems engineering, but only by the engineers using the tools*" (Frosh, 1969). Engineers are people. The People Process Product Time (PPPT) framework or model (Chapter 3) emphasizes effective people (Covey, 1989) since it is people working within the context of an enterprise framework (system) who build a product (the system) over a period of time. The most significant factor in the PPPT approach is the recognition that cost reductions (improvements) in the product and process do not occur in a vacuum (Kasser, 1995). The product under construction is a system and the process producing the product is a system (Martin, 1996). Thus, the process, product and organization represent three tightly coupled systems or dimensions of Quality and must not be considered independently. In addition, every one of these systems changes over time. The PPPT enterprise framework:

- Applies systems engineering to the organisation.
- Is a control and information system paradigm rather than a production paradigm.
- Views the enterprise as a process as shown in Figure 3-2 from the perspective of Information Systems, the application of Knowledge Management, and modern Quality theory.
- Has explicit emphasis on Configuration Management and building Quality into the process and hence into the enterprise.
- Combines prevention with testing and is based on the recognition that prevention is planned anticipation (Crosby, 1981).
- Is used within an organizational engineering or integrated product-process and management paradigm (Kasser, 1999).

From the PPPT perspective, systems engineering is a time-ordered sequence of activities in a multi-threaded environment managed by the CCB[57]. PPPT combines prevention with in-process testing in a synergistic manner to eliminate defects and so reduces project cost and schedule overruns. The PPPT task management methodology:

- Emphasizes teamwork and customer involvement.
- Is loosely based on a methodology used for at least eight years in a task-ordered environment by a large contractor to NASA.
- Improves on the basic methodology by adding the elements of Quality. The improvement:
    - Ensures work is performed in a cost-effective manner.
    - Maps very well into managing tasks performed in geographically distributed locations by different elements of a distributed organization.
    - Intrinsically incorporates task management into program management.
- Builds the Quality into the task.
- Reduces the cost of doing work.
- Allows the needed staffing levels and skill-mix to undergo the gradual change required to perform the planned work in an optimal manner as tasks progress through their lifecycle.
- Monitors task and contract performance relative to the baseline plan.
- Develops measures of effectiveness of the work.
- Incorporates control functions that effectively deal with deviations from the baseline plan in a timely manner.

[57] See Chapter 13.

Deming wrote "*Improvement of quality and productivity, to be successful in any company, must be a learning process, year by year, top management leading the whole company*" (Deming, 1986). Drucker discussed learning organizations as organizations in continuous change (Drucker, 1995). PPPT includes:

- **Continuously monitoring and improving the task:** Training before doing, and applying lessons learned on one project to the next (the feedback loop). Prevention and continuous improvement are important elements of the MBNQA.
- **Making the Technical Performance Measurements:** Supplying the standards and controls for the current task to provide:
  - Visibility of actual vs. planned performance.
  - Early detection or prediction of problems which require management attention.
- **Managing changes:** Supporting the assessment of the program impact of proposed change alternatives.
- **Acting as the advocate for the customer:** During the design and test phases of the task and whenever the customer is not present.
- **Performing Risk Management:** Identifying and mitigating risks to future tasks.
- **Tracking implementation:** Allowing the Program Manager to ensure that tasks are completed on schedule.

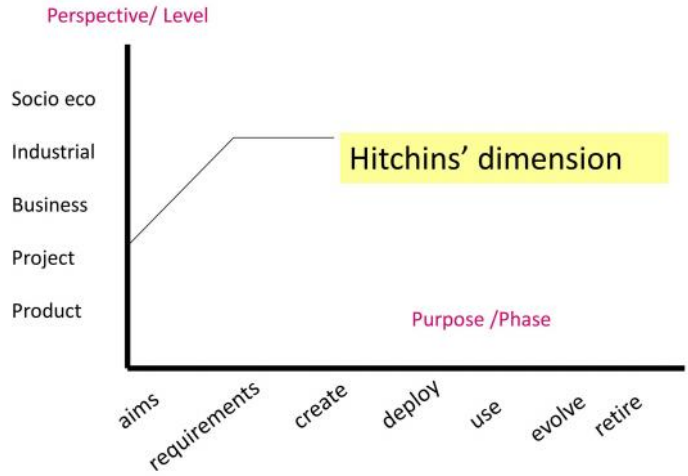


Figure 12-1 Proposed 2D Hitchins-Kasser-Massie SEBoK framework

12.2 A framework for the SEBoK

Each of the definitions and models provide a different insight or view. Perhaps the best approach to develop a framework is a blend. The Hitchins five-layer model provides a useful basis for illustrating how each layer "lives within", and contributes to, the one above and how enablers and constraints at one layer affect the lower layers. However, a broader[58] perspective is needed for a more complete SEBoK. Now if the Hitchins five layer model is enhanced by adding the second dimension of the phase in the SDLC or time, it becomes the Hitchins-Kasser-Massie framework for understanding systems engineering (HKMF) shown in Figure 12-1 [59]. The resulting matrix allows one to provide both the perspective (i.e. layer) and purpose (i.e. phase) for each activity needed in system engineering and hence a justification for why it is needed. For example, each of the Hitchins' layers contains different processes needed to look at the overall aims, obtaining the requirements, creating[60], introduction into service, using, evolving, and retiring a system.

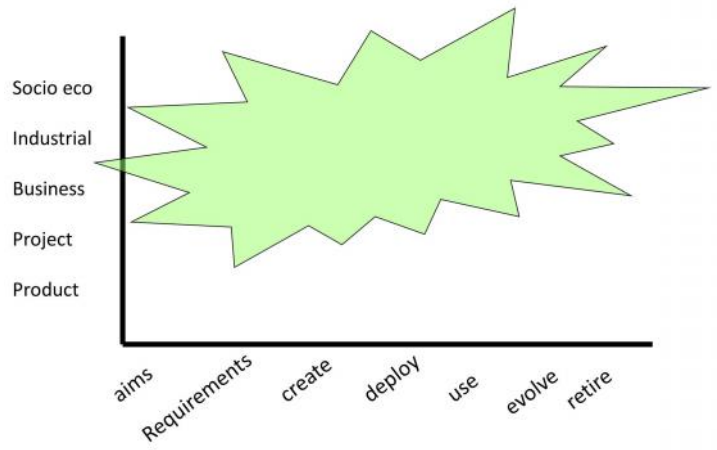


Figure 12-2 Badaway's place in the frame

When the frameworks and models discussed in Section 12.1are placed in the HKMF, it can be seen that:

- Badaway's framework focuses much more on the upper layers of Hitchins' model over the SDLC as shown in Figure 12-2.

[58] Layers 3-5 are often regarded as "management" rather than "engineering" and as such in the academic world are taught via MBA courses.
[59] And redrawn some years later by Ms. Xuan-Linh Tran as Figure 21-3.
[60] Creating includes design, build and testing.

- Sage's model straddles the HKMF as shown in Figure 12-3 advancing up the layers as the SDLC progresses over time.
- Checkland's 's soft systems approach fits into the HKMF in the higher layers as the system is being developed and used as shown in Figure 12-4 (Checkland, 1991).
- The various activities performed in the realisation of systems can tentatively be mapped into the HKMF as shown in Figure 12-5.

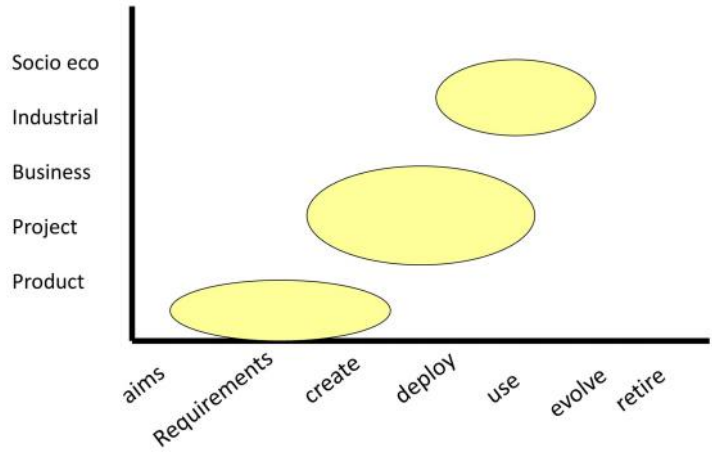


Figure 12-3 Sage's place in the frame

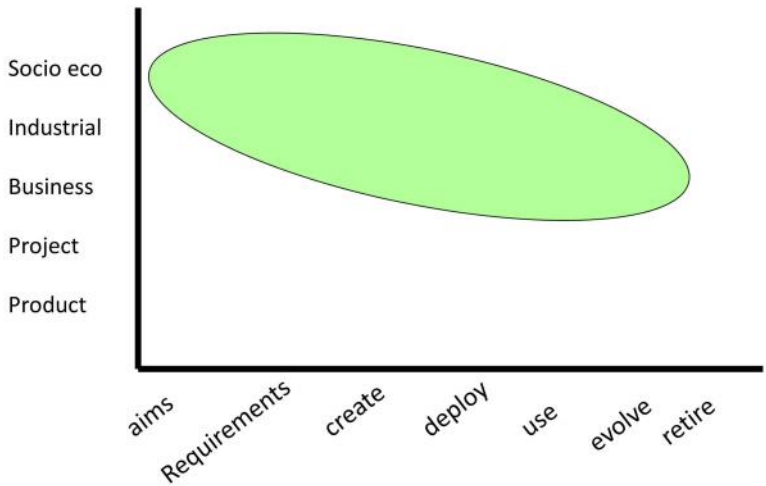


Figure 12-4 Checkland's place in the frame

- INCOSE's current focus seems to be as shown in Figure 12-6.

- The PPPT model which describes the product in the context of the process, people and organization and the changes over time as the product is constructed allows more dimensions to be added to Hitchins' layers. By combining the PPPT and Hitchins' models into a multi-dimensional framework, it should be possible to identify appropriate knowledge, standards, methodologies, skills and competencies for each layer within the framework of products, processes, people and organisation. The information appropriate to each layer should be identifiable from experience and verified from the literature.

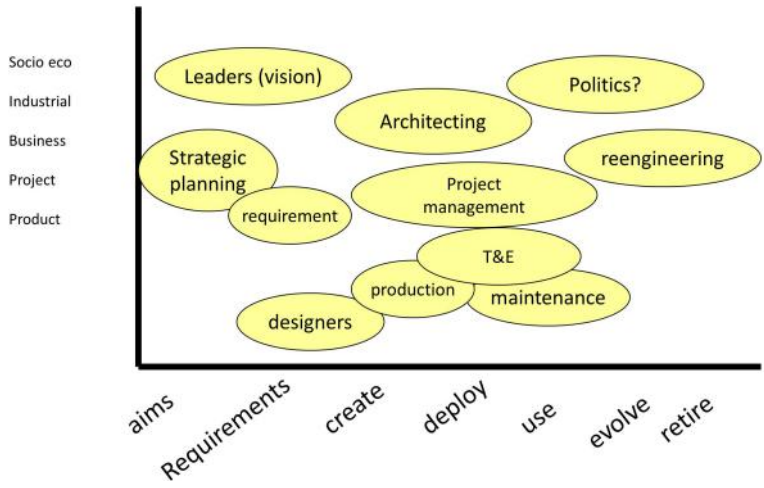


Figure 12-5 Mapping activities into the frame

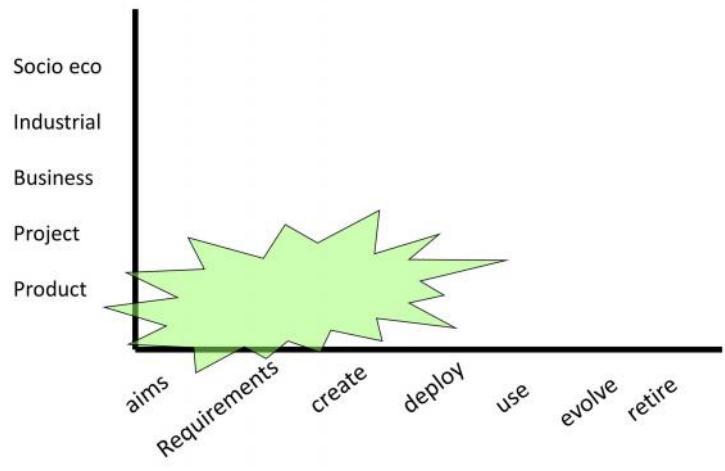


Figure 12-6 Current focus of INCOSE mapped into the frame

Development of the draft SEBoK should take the following form. For each layer in the HKMF, the following should be done in an iterative manner:

- Develop a glossary of terminology. The glossary is to contain references to the source of the use of a word within a given context.
- Develop a concept of operations of the activities being undertaken by means of scenarios or Use Cases.
- Based on the problem solving activities, identify the knowledge needed and appropriate methodologies for cost effective work.
- Develop the vertical interfaces to adjacent layers in the model.
- Identify the horizontal interfaces to adjacent professions to determine the degree of overlap between systems engineering and other professions.
- Search the literature to provide sources for the knowledge in the layer.
- Document the SEBoK using the software engineering body of knowledge as a guide in a similar manner to the way that the systems engineering CMM was developed from the software CMM.

Each layer of information then needs to be verified and validated. To ensure the appropriateness of the information in each layer, representatives from industry and academia should verify the information. Once the first draft of the SEBoK has been compiled it should be used to develop a curriculum for postgraduate education in systems engineering. Then the knowledge has to be mapped into subjects and appropriate instructors found. Due to the broad nature of the material, it can be expected that no single academic institution will have expertise in all subjects. Thus while they may be able to teach them, Industry would be better off if all the subjects were taught by experts. Some kind of teaming arrangement using distance education techniques would provide the optimal program (Kasser, 2000b).

12.3 Perspectives from the HKMF

Even now with a rudimentary HKMF in place, several causes of confusion can be identified. For example:

- **Semantic confusion** – a word may have a different meaning in a different layer. For example, the term “capability” has different meanings in Layers 1 and 3.
- **Role confusion** – systems engineers perform different functions in the different phases of the different layers of the HKMF.
- **Traceability of requirements** – requirements on a system in Layer 1 can be traced back to the socio-economic situation in Layer 5.

12.4 Summary

The advantages of using the HKMF for assembling the SEBoK include:

- Provides a Rosetta stone for communications between different systems engineers. This should avoid the need to continually redefine terminology.
- Provides visibility into what systems engineers actually do.
- Identifies differences in methodologies and tools between hard and soft systems activities.
- Identifies skills and hence training needed at each layer.

2002

13 The cataract methodology for systems and software acquisition

This Chapter views the organisation from the process perspective and discusses in detail the Cataract Methodology for systems and software acquisition (Denzler and Kasser, 1995). The goal of the Cataract Methodology is to achieve convergence between the customer's needs and the operational system as depicted in Figure 5-3. The Cataract Methodology:

- Is based on the recognition that the "Build" approach used in the operations and maintenance phase of the SLC for software maintenance is also applicable to the initial development phase of the SDLC.
- Has been constructed out of components in existing methodologies, each of which have been shown to be effective.
- Extends the spiral approach by emphasizing the criticality of Configuration Management and the type of information needed to control system and software development in an integrated engineering and management environment.
- With its focus on configuration and knowledge management can produce systems that converge with the needs of the customer with fewer cost and schedule escalations and project failures provided appropriate knowledge management and configuration tools are used.
- With its short iterative lifecycle is well suited for agile systems development.

The current systems and software acquisition paradigm is characterized by project failures and cost and schedule overruns. Data from the USA (CHAOS, 1995) and the UK (OASIG, 1996) show that the problem is an international one. Conventional wisdom states that the Waterfall approach does not cope well with changing requirements. Thus efforts to overcome the problem have reacted to the effects of poorly articulated and changing user requirements during the development process and have focused on changing the production process from the waterfall approach to some type of rapid, spiral, or other methodology, with some improvement. Now, from an information systems and Knowledge Management perspective these acquisi-

tion programs do not fail because the requirements change, they fail because of poor requirements management, namely the failure to manage the changing requirements. This Chapter analyses the system and software methodology, provides some insight into the nature of the process generally thought of as being represented by Figure 5-1. The customer has a need that is documented in a SOW and a contract is awarded for development of a product or system that meets the need. The contractor then develops the product over some period of time. There are a number of milestone reviews along the production process to attempt to verify that the development contractor is producing the correct system.

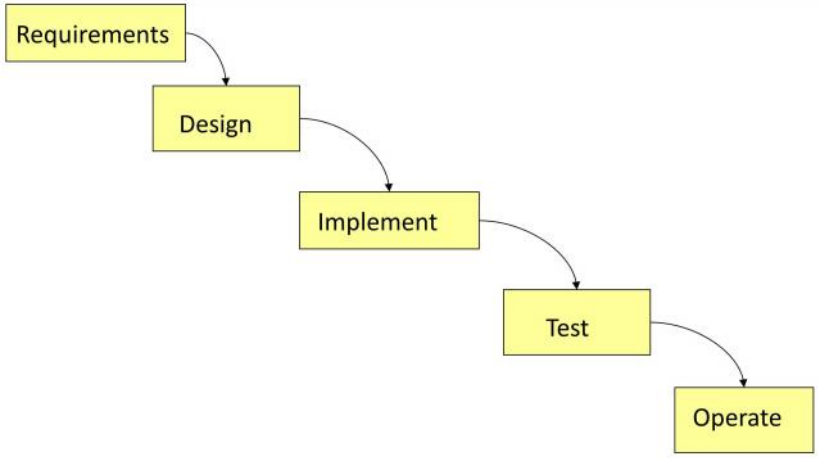


Figure 13-1 The Waterfall methodology

The Waterfall methodology (Royce, 1970) shown in Figure 13-1 was among the first attempts to document the software production process. It represented the process as a serial sequence of events.

The requirements analysis phase is the phase in which the user needs and constraints are determined. The user's needs are then translated into system requirements, which these days are stored in the database of a Requirements Management tool. The process of accepting the initial set of requirements may be represented as shown in Figure 8-2. Each requirement must be considered as a request until accepted and is allocated an identification number. The requirement must be assessed for priority, and cost and schedule impact, as well as for risks. However, during the pre-System Requirements Review (SRR) period, some of these assessments are currently generally not performed. The initial requirements must be considered as not being firm until all the initial system requirements have been documented. During this process the customer and developer must resolve conflicts in the requirements. At that time, the process of gathering the initial set of

requirements generally terminates with the SRR in which both customer and development contractor accept the requirements and the requirements are frozen (no further changes allowed). The customer agrees that the requirements represent the needs, and the development contractor agrees to produce a system that meets the requirements.

The next phase in the waterfall methodology is the design phase, which follows once the requirements have been accepted. It is the phase in which modules of the system are designed to meet the requirements. The implementation phase in which the system is constructed then follows. Once the system is constructed it is formally tested and finally delivered to the customer for use.

Milestone reviews take place between the phases to confirm that the work allocated to a specific phase is complete and the process is ready to advance to the next phase. The name of the methodology was adopted because the pictorial representation shows each phase seeming to flow naturally into the next phase like water flowing over a series of falls.

The Waterfall process is ideal when the vision of the product exists (all requirements are known) at the time that the contract is awarded, and the contractor just builds it. However, in the real world the situation is different as shown in Figure 5-2. During the time that the contractor advances towards the vision of the product that existed at the time the contract was awarded, the vision itself changes. In other words, the target is moving. Thus, while the delivered system may meet its original requirements, the system will not meet the "new" requirements in effect at the time of delivery. The target moves for various reasons including:

- The customer requirements change over time for various reasons.
- The customer has non-articulated requirements at the start of the process and manages to articulate them as time passes.
- Externally driven changes such as changes in government regulations, changes in the marketplace, changes in technology, and changes in other systems that interface with the system.

This situation leads to poorly controlled construction as represented by the chaotic waterfall model shown in Figure 13-2. The figure also shows that the cost to make changes grows the further down the waterfall the changes are made. While the Spiral model (Boehm, 1988) emphasizes risk management, and facilitates the articulation of requirements, it does not emphasize configuration control. If the spiral model is opened up, it can be seen to be a waterfall. Thus while the Spiral model provides some improvement over the waterfall model, lack of configuration control tends to result in moving baselines, chaos and confusion, which lead to cost escalation and schedule delays.

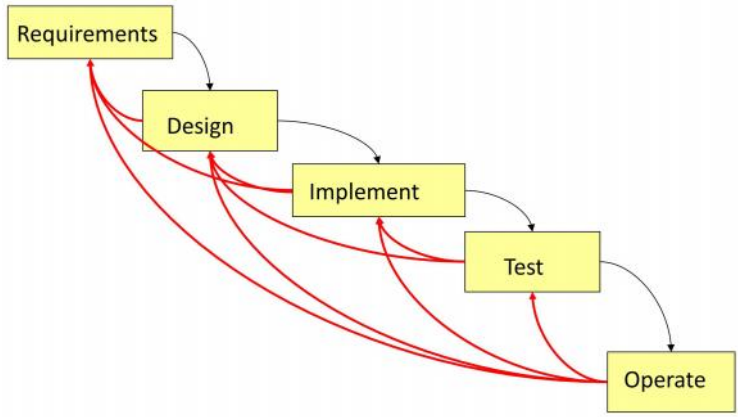


Figure 13-2 The chaotic view of the waterfall

13.1 Budget tolerant Build planning

The initial design is frozen, baselined and presented at a Preliminary Design Review (PDR). In the budget tolerant approach, this particular review differs from the traditional PDR in that lifecycle cost estimates and requirements priorities are included in the design trade studies using the appropriate elements of the QSE (Chapter 8). A detailed design-to-cost development is then initiated, where the highest priority requirements are selected for inclusion in the early Builds until the sum of their costs to implement is within the appropriate margin of the total allowed cost. In this design exercise, the

- Cost of all selected requirements is computed for the entire lifecycle of the system.
- The most necessary requirements are those selected for implementation.
- Builds are organized so that the most critical requirements are implemented first.

Once we start building the system, change management becomes more complicated because the impact of a change can impact portions already built, as well as cause redesign of yet-to-be-implemented requirements. When a change request is made, the systems engineer performs an impact assessment as described in Section 8.1. The priorities of the requirements and the major cost drivers are known being stored in the QSE also described in Chapter 8, so change management means making informed decisions about the following two types of changes:

- Budgetary changes.
- Requirements changes.

13.1.1 Budgetary changes

In today's systems engineering environment, budgets are decreasing while needs are remaining constant or even increasing. Budget changes lead to changes in performance and vice versa. These factors are two sides of the same coin, yet this very simple linkage does not seem to have been made to date. As a matter of fact, the traditional development philosophy tends to keep the cost information isolated from the people who set requirements. One purpose of systems engineering is risk mitigation, yet mitigating the risks introduced by a budget decrease tends to be ignored in the SDLC.

The effect of a budget decrease is change. Some functionality will have to be given up, i.e., requirements will have to be deleted. The change may take two forms:

- cut a certain amount of money from the program, which directly affects the system engineering process within the organization with ramifications on the staffing and schedule, and
- cut some requirements from the system under development, which has a direct impact on the product and an indirect impact on staffing and schedule.

The way to deal with budgetary changes is to identify the lowest priority requirements. Assess the impact of deleting them. Sometimes work already completed may change absolute costs. Then delete the lowest priority requirement(s) consistent with the budget reduction from future Builds. The low priority requirements should have been assigned to the later Builds to facilitate this.

13.1.2 Requirements changes

The real world of continuously changing requirements is recognized by the statement that the goal of system engineering is to provide a system that (Kasser, 2000c):

- Meets the customer's requirements as stated when the project starts.
- Meets the customer's requirements, as they exist when the project is delivered.
- Is flexible enough to allow cost effective modifications to be implemented as the customer's requirements continue to evolve during the operations and maintenance phase of the system lifecycle.

This is of course impossible with today's technology. However, in many projects it may be possible to come close and achieve a large degree of convergence between the requirements and the capability of the system. The major lesson learned seems to be not to identify all the requirements at the start of the project, but to identify the (Kasser, 2000c):

- **Highest priority requirements** - The show stoppers. The risk here is the failure to identify the critical requirements and the failure to set the priority correctly.
- **Real requirements** - As opposed to apparent requirements.

Since the requirements change over time, there is a need for Build planning. The system must be built in such a manner that the requirements are implemented in the order of their priority (Denzler and Kasser, 1995). The design path takes one from the domain of where everything is possible to what is actually possible. Thus:

- **Detailed design decisions should be made on a JIT basis** (Kasser, 2000c). There is no need to complete the design before starting a Build. However, the design must be feasible. The risk here is in determining the feasibility of the design. For example, in a case where the need is for synchronous voice communications between two places. Since an initial assessment shows that the need can be met using the conventional telephone service or by the use of voice over the Internet, there is no need to make that decision early in the design cycle. The characteristics of the telephone link are known. The characteristics of Internet voice links are also known. Experiments can take place and the actual decision made just in time to implement the communications links. Since there is a possibility that the requirement for synchronous communications may be deleted in the future, any design effort made earlier would be wasted if the requirement were eliminated. In addition, if the requirement is not eliminated, then advantages can be taken of improvement in technology and/or cost reductions over the time before the decision has to be made.
- **Design decisions must also maximize the "don't cares" as well**. The example here is Internet voice works (risk minimal) but the actual choice of how to implement the communications subsystem can wait for a while. A better example is from the LuZ SEGS-1 sun sensor glue case (Chapter 11). If the requirement had been placed on the process, not to allow glue on the face of the diode, the characteristics of the glue under the high temperature conditions would not have mattered and the expensive sun-sensor replacements would have been avoided.

Thus the key to an effective SDLC is to manage change in a manner that achieves convergence between the needs of the user and the capability of the as-built system in a cost-effective manner as shown in Figure 5-3. The way to achieve this goal seems to be not to attempt identify all the requirements at the start of the project, but to only identify the highest priority and the riskiest-to-implement requirements. Then to achieve convergence by fleshing out the requirements in a controlled manner and delaying design decisions using a JIT approach (Kasser, 2000c) in the Cataract Methodology.

13.2 The Cataract methodology

The Cataract Methodology relies on two factors:

- The waterfall methodology works very well over a short period of time.
- Implementation and delivery of systems and software are often performed in partial deliveries, commonly called "Builds" in which each successive Build provides additional capabilities.

Build planning is not a new concept. It has been used in software maintenance for many years. It was also incorporated in the UK Defence Evaluation and Research Agency (DERA) Reference Model (DERA, 1997) shown in Figure 13-3.

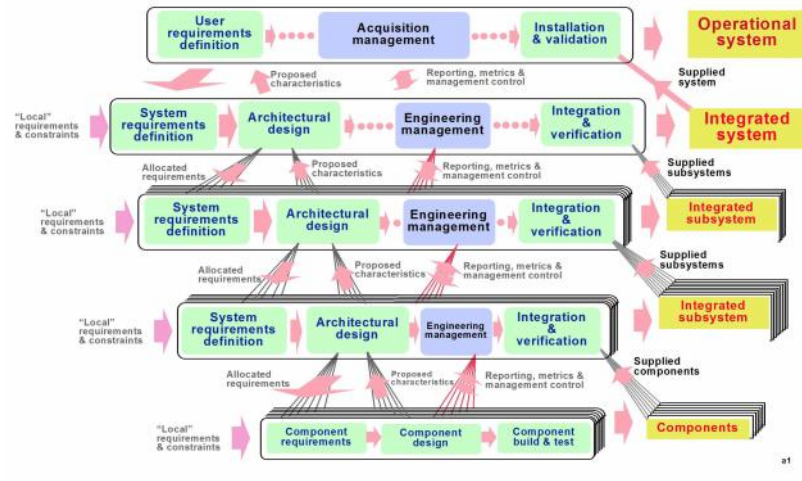


Figure 13-3 DERA evolutionary lifecycle (DERA, 1997)

In the software world, a Build means a defined software configuration. Successive Builds enhance the capability of the software. In the hardware world, Builds can comprise subsystems, or the integration of two or more subsystems. The work associated with each Build takes place in the three parallel streams of activities shown in Figure 2-2 and is organised such that:

- **Systems engineering** performs requirements and interface engineering, change assessment, risk management, allocates the system level requirements between the hardware and software components, coordinates technical performance analysis and measurement, and produces the process-products (documentation).
- **Software engineering** turns the software requirements into software code, and evaluates and perhaps incorporates COTS software.

- **Hardware engineering** may be working with the computers, workstations, disk drives or other storage elements, networks, and custom hardware elements.
- **T&E** develops test plans and procedures, then performs the tests and reports on the results.
- **System Integration** integrates the hardware and software units and verifies their working together.
- **Final testing** in which the integrated Build is tested prior to acceptance by the customer.
- **Transition** is the time in which the Build is turned over to the customer or user.
- **Operations and maintenance** is the time span when the Build is operated by the customer, or by the maintenance contractor.
- **Management** is the planning, organizing, directing, and controlling the technical and administrative work. This includes making sure that the needed resources are available at the appropriate time.

The cataract approach to Build Planning may be likened to a Rapid Prototyping scenario within the spiral in which the requirements for each Build are frozen at the start of the Build. This approach, however, is more than just grouping requirements in some logical sequence and charging ahead. Build plans must be optimised on the product, process, and organization axis to:

- Implement the highest priority requirements in the earlier Builds. Then, if budget cuts occur during the implementation phase, the lower priority portions are the ones that can readily be eliminated because they were planned to be implemented last.
- Make use of the insight that, typically, 20 percent of the application will deliver 80 percent of the capability (Arthur, 1992) by providing that 20 percent in the early Builds.
- Allow the waterfall approach to be used for each Build. This tried-and-true approach works on a small project over a short timeframe.
- Produce a Build with some degree of functionality that if appropriate can also be used by the customer in a productive manner. For example, the first Build should generally, at a minimum, provide the user interface and shell to the remainder of the functions. This follows the rule of designing the system in a structured manner and performing a piecemeal implementation.
- Allow a factor for the element of change.
- Optimise the amount of functionality in a Build (features versus development time).
- Minimize the cost of producing the Build.

- Level the number of personnel available to implement the Build (development, test, and systems engineers) over the SDLC to minimize staffing problems during the SDLC.

13.2.1 Build zero

Once the initial set of requirements has been signed off, the system architecture designed, and the implementation allocated into a series of Builds, the implementation phase embodying the cataracts begins. The Cataract methodology incorporates an initial Build, Build Zero, which contains the same initial two phases, requirements and design, of the Waterfall methodology with the exception that there is recognition that

- All the requirements are not finalized at SRR.
- Additional requirements will become known as the project progresses.
- Design and implementation decisions will be deferred so as to maximize the "don't care" situations and made in a JIT manner (Section 13.1.2).

The work in Build Zero is to:

1. Identify the highest priority requirements.
2. Baseline an initial set of user needs and corresponding system requirements.
3. Develop the FREDIE database incorporating the QSE for each of the baselined requirements (Chapter 8.3). The FREDIE provides the data necessary for making informed decisions about accepting the initial set of requirements and subsequent changes.
4. Complete the first draft of the SEMP and OCD (Kasser and Schermerhorn, 1994b).
5. Design the Architecture Framework for the system. According to the DERA Reference Model, "typically, architectural design involves identifying and exploring several design options at a level of detail consistent with the technical and commercial risks. The options evaluate the performance and attributes of alternative architectural structures. Based on an appropriate level of understanding of candidate components, the feasibility of particular architectures or architectural variants is confirmed and their performance assessed. The trade-off analysis leads to a preferred architecture solution and to confirmation of the requirements for the subsystems/component functions and interfaces necessary to effect that architecture" (DERA, 1997).
6. Perform risk assessment to determine the proposed Architecture Framework can meet all of the highest priority requirements.
7. Document the assumptions driving the Architecture Framework and a representation of operational scenarios (Use Cases) that the Architecture Framework prohibits. This activity also helps identify

missing and non-articulated requirements early in the SDLC. The design of the Architecture Framework for the entire system in Build Zero introduces a risk that it may not be suitable for changes years later in its operations and maintenance phase (or even earlier). This is why part of the Build Zero effort is to determine scenarios for which the system is not suitable. The customer is then aware of the situation. The goal of the Cataract Methodology is to achieve convergence between the customer's needs and the operational system. In the course of time, one can expect that the need will change to something for which the system cannot provide capability. At that time, a revolutionary Build will be needed to replace the system. However, it will be done with full knowledge in a planned manner, rather than the ad-hoc manner of today's environment.

8. Develop the WBS to level the workload across the future Builds and implement the highest priority requirements in the earlier Builds as described above.
9. From Build One inclusive, each subsequent Build is a waterfall in itself. The requirements for the Build are first frozen at the Build SRR. Then the design effort begins. Once the design is over, the Build is implemented and when completed turned over for integration. While the design team does assist with the integration, their main effort is to start to work on the design of the next Build. Once the first Build has been built and is working, the requirements freeze, design - integrate - test - transition and operate stages of the SLC commences for the second Build. This cycle will continue through subsequent Builds until the system is decommissioned although the contract may change from the development organization to the maintenance organization. Each Build is an identical process but time delayed with respect to the previous one. Each successive Build provides additional capabilities. When the Builds are placed under configuration control, the Waterfall may initially be drawn as shown in Figure 13-4 however this figure is misleading. Externally driven changes are requested and problems tend to show up during the integration and test phases. When a problem is noticed, a discrepancy report (DR) is issued against the symptom. This DR is analysed and the cause identified. A change request is then issued by the CCB to resolve the defect either in the current Build before delivery, or by assigning it to be fixed in a subsequent Build. Thus Figure 13-4 should be replaced with Figure 13-5 showing that the Cataract methodology explicitly adds:

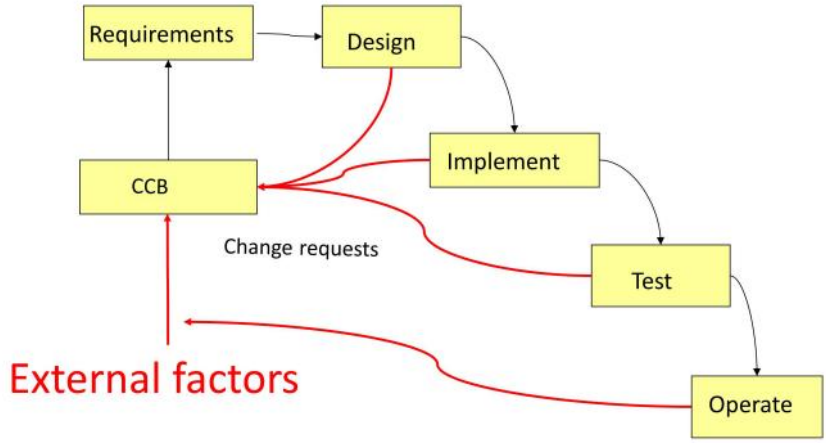


Figure 13-4 Configuration control view of Waterfall

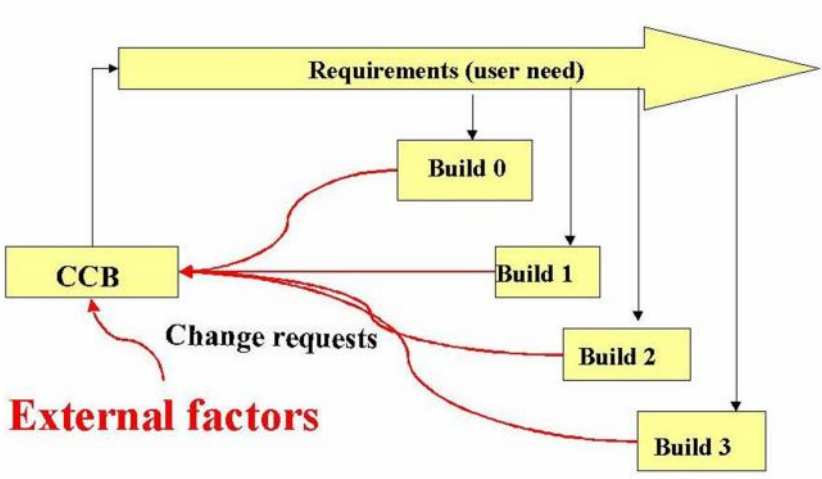


Figure 13-5 The cataract methodology

- the effect of changing external requirements, and
- the management of changing requirements

to the DERA Evolutionary Lifecycle approach Figure 24 (DERA, 1997) shown in Figure 13-3. The change request, if accepted, is assigned to be implemented in the appropriate future Build in the view shown in Figure 8-1. Think of each Build as being completed a little behind the arrowhead of the advancing requirements. From this perspective, the gap between the user's need and the completed section of the system converges over time as shown in Figure 5-3.

Project personnel move from one Build to the next; the development team moves from one Build to the next, as does the testing team. Ideally the Builds are sequential with no wasted time between them. The customers tend to get increasingly involved with the system during later Builds by virtue of being able to use early Builds.

Each Build is placed under configuration control and may be delivered to the customer. Accepted change requests modify the requirements for future Builds, with the sole exception of "stop work" orders for Builds-in-progress if the change is to remove major (expensive to implement) requirements being implemented in a Build-in-progress. The milestone reviews within a Build are identical to those in the Waterfall methodology, since the Build is implemented using a Waterfall. All change requests received during any Build are processed and if accepted are allocated to subsequent Builds. Freezing of the requirements for each Build at the Build SRR means that when the Build is delivered it is a representation of the customer's needs at the time of the Build SRR. It may not meet the needs of the customer at the time of delivery, but the gap should be small depending on the time taken to implement the Build since the SRR, thus achieving convergence between the needs of the customer and the capability of the as-delivered system.

13.3 Contractual boundaries

The insight presented herein is that as soon as the first Build in the development process has begun, the development process and the maintenance process are identical. The requirements freeze, design - integrate - test - transition and operate stages of the SLC commences with the second Build and the cycle will continue through subsequent Builds until the system is decommissioned although the contract may change from the development organization to the maintenance organization. If this is represented in Figure 13-6 it can be seen that the Turnover point between the development contractor and the operations and maintenance (O&M) contractor is an artefact of the boundaries between the contracts awarded by the acquisition organisation. This is because:

1. The effect of change is not generally considered to be in the domain of the development contractor even though the customer's needs are changing during the development time.
2. The work performed by the O&M contractor is to implement requested changes, upgrade the performance, and sustain the system as well as fix defects.

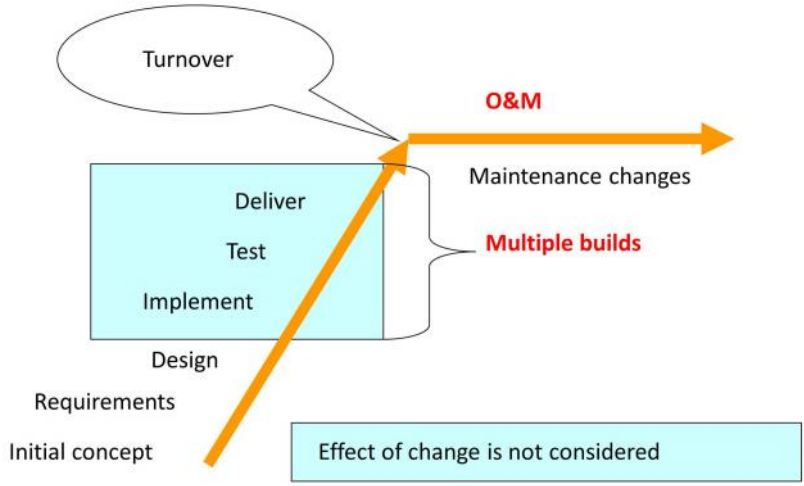


Figure 13-6 The Build cycle view of the SLC

13.4 Changes

The effects of the changes will show up as a variance in the cost and schedule because the impact of a change affects requirements, documents, the WBS, Builds, deliveries, and cost and schedule, depending on the point in the SDLC in which the change occurred. Effects that will be noticed include:

1. Changes in high-level requirements will affect lower level requirements and may affect implementation requirements.
2. Documents affected could include management plans, operations concepts, manuals, test plans, and procedures.
3. WBS elements which contain not include all SDLC activities. All SDLC activities need to be in the WBS.
4. For Builds and deliveries, the implementation sequence may be changed to the point where a Build no longer adds any value to the system, so the cost of testing, releasing, and delivering the Build may no longer be economical.

Donaldson and Siegel state that there are two types of changes during the SDLC namely planned and unplanned (Donaldson and Siegel, 1997). The implementation of each type begins with a change request.

13.4.1 Change requests

The process for dealing with both types of change is the same and begins with a change request. The change requests are processed via the CCB (Chapter 8). Requests for planned changes tend to be processed well before

the change is to be implemented. Requests for unplanned changes however, need to be categorised by priority. Typical categories may be "routine", "urgent", or "do by yesterday" or their equivalents. A typical "do by yesterday" change request is the result of an analysis of a DR reporting that the system crashes.

13.4.2 The generic process for handling a change request

The change request process is the same as the process for accepting the initial set of requirements at the start of the SDLC shown in Figure 8-2. A typical impact of a requested change on the product and process (Builds) shown in Figure 13-7 is assessed and a decision made as to whether to accept or reject the request. The Figure shows the WBS elements arranged as a process flow. This is valid since in the requirements-based paradigm all work is represented in the WBS.

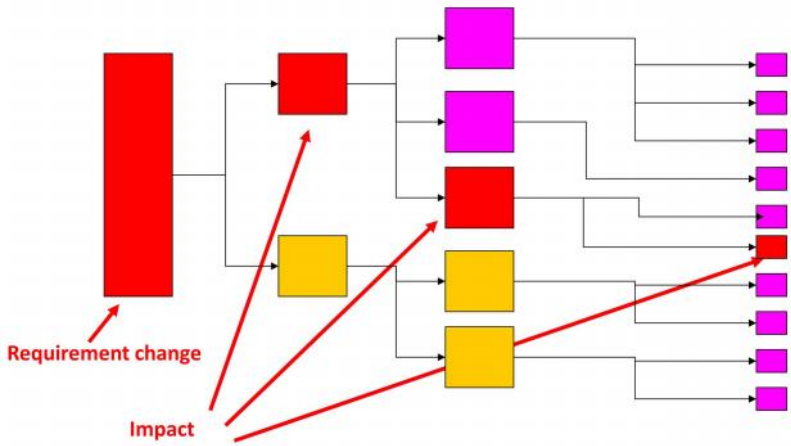


Figure 13-7 The typical impact of a change request on WBS elements

13.5 The configuration control process

Conventional configuration control tends to be limited to products that are either in process of construction or have been completed. However, the key to effective control of the production process is effective configuration control and informed decisions about the impact of any change request on both the product (cost and capability) and the process (cost and schedule). Thus the impact of the requested change on the process as represented by the Build Plan and WBS also needs to be assessed as well as the impact on the functionality or capability of the product under development.

13.5.1 Impact Assessments

The purpose of the impact assessment must be to determine the cost, feasibility and risk of each requested change to both the product and process. The elements of the impact assessment of a change request[61] are the:

1. Determination if the request has been rejected before and if those reasons are still applicable.
2. Determination if the request has been accepted but not yet implemented.
3. Determination if a conflict or contradiction exists with other requirements and if so, resolve it.
4. Determination if the requirement/change is really needed.
5. Determination of the change in the total project risk on the schedule.
6. Determination if the change is feasible.
7. Estimation of the cost to implement the requirement/change.
8. Determination of the cost drivers for the change in the design.
9. Performing sensitivity analyses on the cost drivers. The results of a typical sensitivity analysis are shown in Figure 13-8.

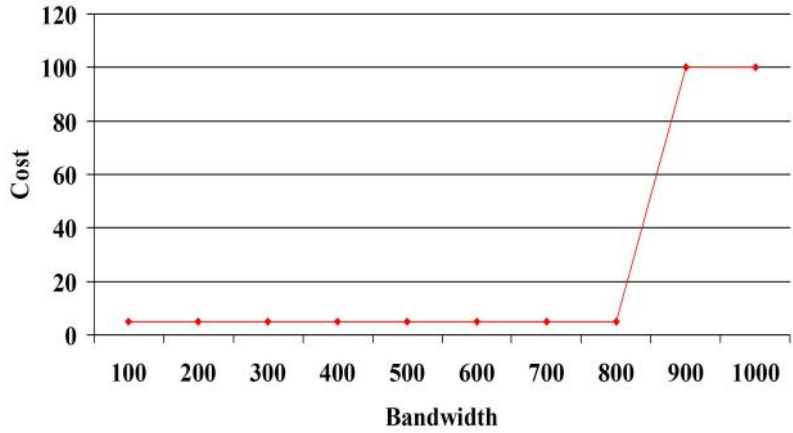


Figure 13-8 Typical sensitivity analysis graph

10. Discussing the cost drivers for the results of the sensitivity analysis with the customer and determining if the cost drivers are really necessary. For example, in a given system, the requirement was for a communications bandwidth of 900 units. If the requirement had been accepted without the analysis, the cost to implement would

[61] Compare these to those for a new requirement discussed in Section 8.1.

have been estimated as 100 units of money. However, when presented with the graph shown in Figure 13-8, the customer was able to state that the bandwidth requirement of 900 had been based on building in excess capability on the real need for a bandwidth of 450. Thus lowering the bandwidth requirement to 800 would still provide more than the needed capability and reduce estimated cost by about 95%.

11. Documenting the decisions in the system requirement repository.

13.5.2 Typical impact assessment questions

Typical impact assessment questions that may be used to guide the assessment are:

- Why do we need the change?
- What if we don't accept it?
- What are the alternatives?
- What will the modified system do?
- How will the change contribute to mission objectives?
- How will the change impact existing and planned adjacent systems?
- What are the risks and their probability of occurrence?
- How easily can those risks be mitigated?
- What are the required resources required for implementing the change?
- How long will it take to implement the change?

13.5.3 The analysis of the change request

The CCB will assign incoming change requests to an Integrated Product Process Team (IPPT). The IPPT will analyse the change and perform the impact assessment to the appropriate depth to minimise risks. The analysis phase of the impact assessment can be considered as the traditional design phase of the SLC. If the change request is considered as a requirement request, then the process of meeting the requirement is the design of alternative solutions and the choice of the optimal solution, namely the analysis and synthesis functions of the SDLC. The analysis is a problem solving and fixing exercise as illustrated in the feedback loop shown in Figure 13-9.

The problem is examined and analysed which results in one or more solutions being proposed. If the root cause of the problem is not found, a solution may not work or may only work for a short period of time. In addition, even if the implemented solution works it may introduce further problems that only show up after some period of time. Consider the implications of the time delay in the problem solving feedback loop. Any action has an effect in the present and in the future. Group these effects in time as:

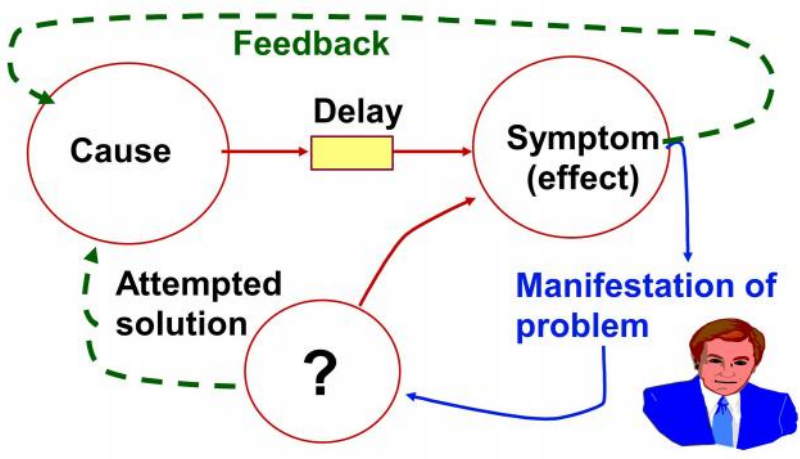


Figure 13-9 Problem solving feedback loop

- **First order** - noticeable effect within a second or less.
- **Second order** - noticeable effect within a minute or less.
- **Third order** - noticeable effect within an hour or less.
- **Fourth order** - noticeable effect within a day or less.
- **Fifth order** - noticeable effect within a week or less.
- **Sixth order** - noticeable effect within a month or less.
- **Seventh order** - noticeable effect within a year or less.
- **Eighth order** - noticeable effect within a decade or less.
- **Ninth order** - noticeable effect within a century or less.
- **Tenth order** – noticeable effect after a century or more.
- And so on into the millennia.

The analysis of the requested change has to consider all of the above as applicable. While the higher order effects may not be applicable in a computer-based system, they are applicable in long-lived systems such as those that affect the environment (Dams, power plants, etc.).

13.6 The Decision

When the impact analysis is complete, the customer's project manager makes the final decision on the effect of a change based on their willingness to pay. Two factors affect the decision, namely the recommendations of the IPPT (based on the result of the analysis) on the impact of the requested change on the cost, schedule and risk, and the willingness of the customer to accept the impact.

13.7 The suite of tools for configuration control

The suite of tools needed for configuration control in the Anticipatory Testing environment are a combination of several existing different and usually unconnected tools (e.g. Requirements Management, Project Management, WBS, Configuration Control, and Cost Estimation, etc.), used in today's management and engineering work streams of the SLC. In addition the tools must also provide the capability to perform:

- impact assessments of the effects of proposed changes before implementation,
- trade studies to compare the costs, risks, and capabilities of alternative designs.

13.8 The Configuration Control Board

The CCB controls change. The acronym can also mean Change Control Board. The CCB must control all changes in the SDLC, those affecting both the process and the product. Thus project management must be a function of the CCB something that is not typically done in the current project management paradigm. The elements of a typical CCB organisation are shown in Figure 13-10. Note that the project manager may be within the contractor or customer organisation depending on the time and place.

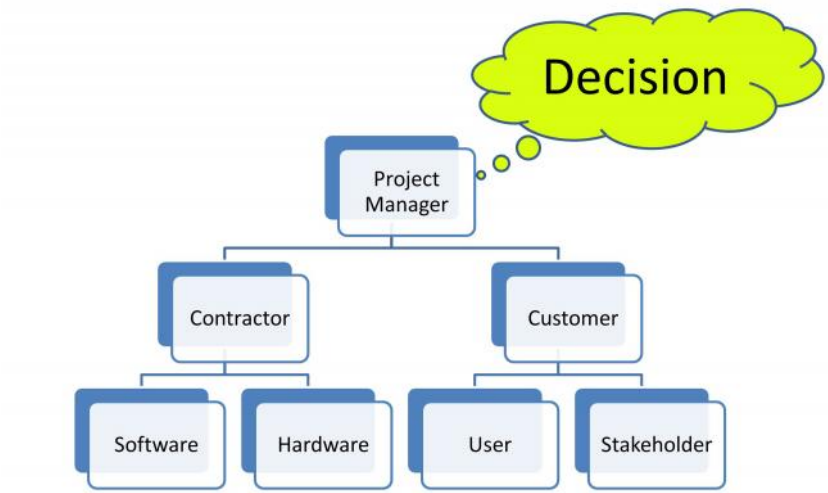


Figure 13-10 Elements of a typical CCB

13.8.1 CCB composition

The customer may have a single representative or several, representing the user and stakeholders on the CCB. In government acquisitions or mainte-

nance contracts the customer representative on the CCB should be the Contracting Officer's Technical Representative (COTR). The COTR knows the total contract budget and feeds the needs of the users and stakeholders to the CCB. The contractor can estimate the risk, cost and schedule impacts. The project manager then makes the decisions within the scope of the contract.

13.8.2 CCB decisions

All changes affect the cost of production. The factors that the CCB considers in making the decisions (accept or reject the change request) are different depending on the type of contract. In a cost-plus arrangement, the costs will be passed on to the customer in the form of additional or reduced costs depending on the nature of the change. In a firm fixed price (or design to cost) contract, if a requested change adds cost, then either the contractor will absorb the cost, the price will be increased, or some functionality will have to be removed to keep the costs fixed. This will probably be a lower priority requirement than the one being changed. In this situation, the CCB must ask the customer to decide which requirement to remove. This situation is one of the reasons for the "priority" element in the QSE.

13.9 Improving the CCB

The CCB for a single project may be improved in several ways including the following:

- Adding T&E capability
- The customer single point interface

13.9.1 Adding test and evaluation capability

In many instances in government contracts, the government employs a separate contractor to perform the T&E or IV&V of the development contractor's work. A more effective CCB is shown in Figure 13-11. Bringing the T&E representative onto the CCB at the start of the project means that:

- The effect on the change on the T&E process will be part of the impact assessment.
- T&E (or IV&V) are brought into the project at the start, unlike in most of the current situations when they are brought in only after the defects have been designed into the system. This difference, in itself, is a major lifecycle cost-reducing element, by allowing interdependent in-process testing and early detection of defects, i.e., the Anticipatory Testing concept.

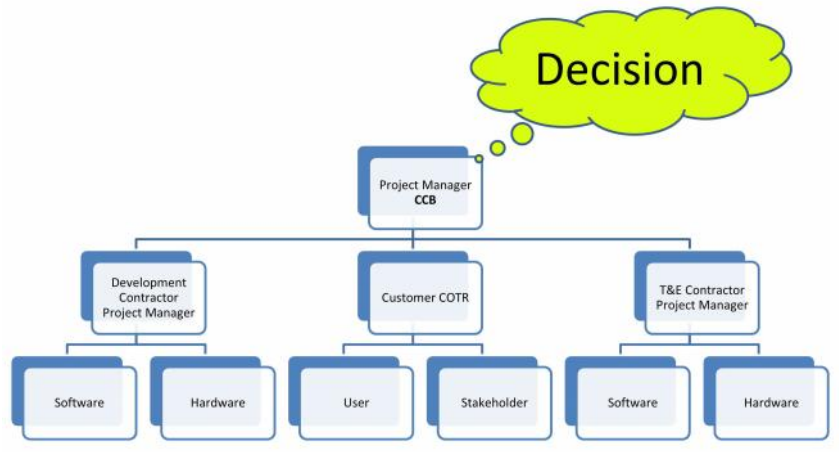


Figure 13-11 A more effective CCB

13.9.2 The customer single point interface

Figure 13-11 shows the COTR as a single interface to the customer. The contractor thus has only one customer to satisfy so there is a clean interface. This does not mean that the users and stakeholders do not need to be satisfied. The task of satisfying them remains wholly within the customer's organisation. This organisation structure provides only one person with the authority to authorise changes. Users and stakeholders however may still be present in the CCB as part of the IPPT.

13.10 Lifecycle implications

Consider the implications on the lifecycle.

13.10.1 The recursive lifecycle

The change request process is the same as the process for accepting the initial set of requirements at the start of the SDLC shown in Figure 8-2. Thus the only difference between a requirement at the start-up phase and a change sometime later in the entire system lifecycle (SLC) is that a start-up is a transition from no system to some system, while a change is a transition from some system configuration to a different system configuration. Thus from this perspective the traditional design process is the impact assessment of the cost, schedule and risk of meeting the customer's need of a number of designs and the choice of the optimal design to meet the initial set of requirements. As such, the lifecycle phases of the SLC are recursive. For example the feasibility analysis performed at the start of the SLC is functionally identical to the design processes later on, the difference being in the amount

of information available at the time, the depth of the analysis, and the terminology used.

13.10.2 The type and place of the IPPT in the SLC

There are two types of IPPT in the SLC, permanent and temporary. Each contains the appropriate knowledge and skills necessary to complete its assignments. There are also two places for an IPPT in the SLC. The CCB is permanent IPPT and the impact assessments of change requests are performed by temporary IPPTs.

13.11 The cataract perspective

From the cataract perspective:

- Successive Builds do not have to be incremental or evolutionary, they can also be revolutionary, i.e. an entire replacement system can be factored into the schedule. Thus legacy systems can be upgraded and replaced with minimal waste of resources using the Cataract Methodology. By knowing when parts of the system will be replaced (in which Builds), informed decisions can be made as to which defects to fix, which modifications to make to the current system and which modifications to defer to the replacement system.
- The traditional concept of developing the functionality of a system by making one change at a time uses the waterfall approach via a sequential series of waterfalls, namely cataracts.
- The Year 2000 issue was just a DR and changes made as a result of the analysis of the problem.
- Effective configuration control and information about the state of the project is vital.

The Cataract methodology depends on a new generation of tools and information displays such as the QSE, FREDIE, and CRIP Charts. The Cataract Methodology is an integrated product-process (engineering and management) methodology that can be used to control costs and schedules and minimize project failures.

13.12 Summary

By viewing the SDLC from the perspective of Builds it can be seen that

- The SDLC is a time-ordered series of tasks. In addition, since the development contractor may be working on more than one Build at a time, each Build being in a different part of its SDLC, the total SDLC is also a parallel process with phase-delayed elements similar to the representation in Figure 11-2.

- Except for Build Zero, the type of work performed in the SDLC, namely up to the time the development contractor turns the system over to the customer (and the maintenance contractor) is identical to the work performed during the O&M phases of the SLC.

13.13 Conclusion

Both the SDLC and the SLC are multi-phased, time-ordered, parallel-processing tasks. The Cataract Methodology with its focus on configuration and knowledge management can produce systems that converge with the needs of the customer with fewer cost and schedule escalations and project failures provided appropriate knowledge management and configuration tools are used.

2002

14 Managing Systems of Systems

This Chapter provides an alternative perspective to much of the research effort being expended in an effort to develop new concepts that can be used to solve the problem of managing Systems of Systems. This Chapter views the product and process from the alternative perspectives shown in Figure 1-3 and shows that from an information and knowledge management perspective the SLC for a single system is a multi-phased time-ordered parallel-processing recursive paradigm that is little different from the uncoordinated ad-hoc evolution of what has been defined as a System of Systems. Hence, after providing the necessary coordination, information and knowledge management based tools and techniques may be used to control the SLC of each of the individual systems in the System of Systems as well as the System of Systems itself.

14.1 Introduction

The term System of Systems in its permanent sense (Cook, 2001) has come to mean a set of interdependent systems evolving at different rates, each at a different phase of their individual SLCs. Controlling the evolution of a System of Systems is deemed to be a complex problem since the development and acquisition paradigm for even a single system is characterised by cost and schedule overruns, and project failures (Chapter 3 and 6). Thus much research effort is being expended in an effort to develop new concepts that can be used to solve the problem of managing Systems of Systems. This Chapter shows that the problem is not as complex as it appears, and is solvable at relatively low cost, if viewed from a different perspective.

14.2 One System

Consider one new system within the System of Systems. The SDLC can be represented by Figure 14-1, which shows a sequential process producing the product. The system level requirements are the combination of the user's needs and external factors such as changes in technology, regulations, and

other factors that affect the system. Since all systems being acquired interact with adjacent systems, some of the external factors come from those adjacent systems.

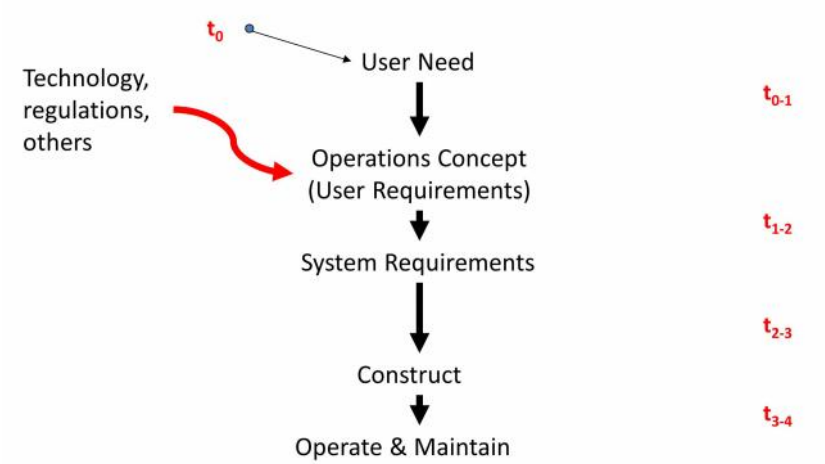


Figure 14-1 The static system acquisition

However in the real world changes take place throughout the SDLC as well as during the operations and maintenance phase of the SLC as a result of internal and external events. This situation may be represented as shown in Figure 14-2.

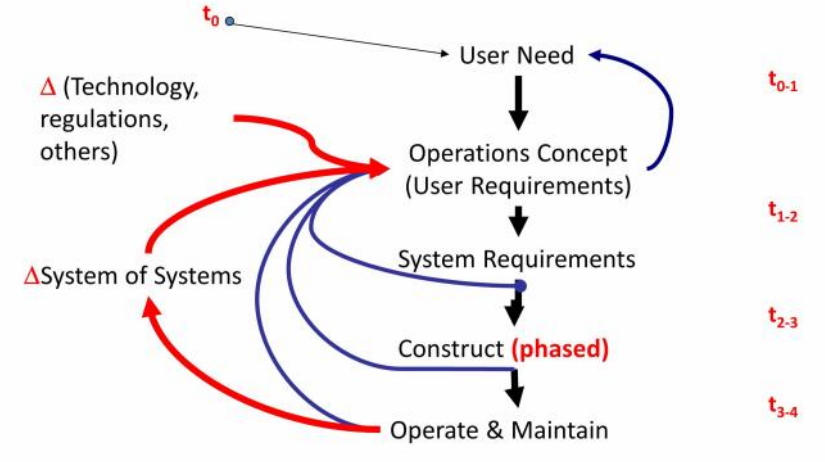


Figure 14-2 The dynamic system acquisition

14.3 The external perspective

Consider the system within the context or framework of its adjacent systems (Chapter 11). The context diagram is shown in Figure 11-1. The system has an interface with several other systems but not necessarily all the systems in the framework. The temporal perspective of the evolution of the same set of systems within the framework is shown in Figure 11-2. Each horizontal line represents an evolving system. The implementation and delivery of such systems and software are often performed in partial deliveries, commonly called "Builds" in which each successive Build provides additional capability (Chapter 13). Thus the sequential blocks in Figure 11-2 represent the operational phase of the different Builds within the individual system's SLC. The SDLC phases of the Build have been abstracted out to simplify the figure. Lines begin when new systems are brought into existence, and terminate when existing systems are decommissioned.

This is the almost the same representation as that for a System of Systems. The difference being that the System of Systems is evolving in an ad-hoc uncontrolled manner. Thus controlling the acquisition of a System of Systems is a matter of identifying the framework for the System of Systems and then managing the evolution of every system in the framework making optimised decisions for the framework as a whole. This is the standard systems engineering methodology for any set of subsystems in a single system. So from this perspective, three problems need to be solved to solve the System of Systems problem, namely:

1. Develop a cost-effective SDLC for a single system that:

- can take into account the on-going changes in adjacent systems,
- meets the customer's requirements as stated when the project starts,
- meets the customer's requirements as they exist when the project is delivered, and
- Is flexible enough to allow cost effective modifications to be implemented as the customer's requirements continue to evolve during the operations and maintenance phase of the system lifecycle (Section 9.2).

2. Implement the links between the systems so that changes in the adjacent systems are taken into account in the evolution of each of the systems.
3. Provide the management information system to make optimal decisions for each system and well as the System of Systems (Hitchins, 1998).

14.4 The cost-effective SDLC for a single system

The Cataract Methodology (Chapter 13) with its focus on configuration control and knowledge management can produce systems that converge with the needs of the customer with fewer cost and schedule escalations and project failures provided appropriate knowledge management and configuration tools are used. Therefore if the Cataract Methodology is applied to every individual system in the System of Systems **and** to the System of Systems as a whole:

- The SLC of each of the systems will be managed in a cost-effective manner.
- The interface for links between the CCB of each and every system will exist via the "external factors" input element.
- The evolution of the System of Systems as a whole will change from an ad-hoc manner to a coordinated manner.

14.5 Another perspective on the system of systems

If Figure 8-1 is redrawn in the format of Figure 11-2 it will look like Figure 14-3. A combination of Figure 11-2 and Figure 14-3 is shown in Figure 14-4. Each of the individual projects has its own CCB operating as (Chapter 13). Figure 14-4 shows a Strategic CCB (SCCB) operating in a similar manner to the CCBs for the individual systems. The SCCB interfaces to all the systems within the System of Systems and also has an external interface.

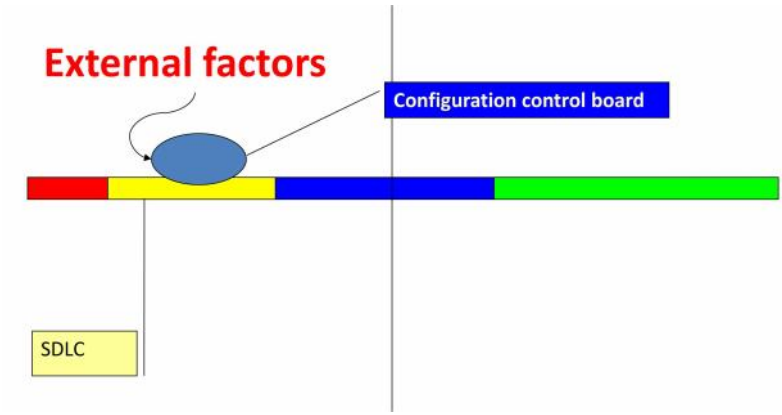


Figure 14-3 System with CCB

14.5.1 The parallel system lifecycle

When Figure 11-2 and Figure 13-5 were combined into Figure 14-4, the Builds were flattened into a series of rectangles and the CCBs brought out and shown as a circle. As such this drawing can be applied to the develop-

ment of subsystems within any of the individual systems. All subsystems might be developed in-phase in some of those cases. However, the normal approach of completing one subsystem at a time and then integrating it to the others is a multi-phased activity.

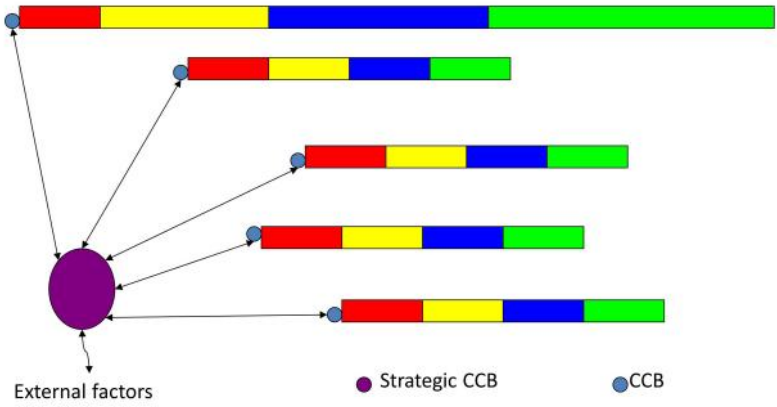


Figure 14-4 Controlled phased parallel evolution

The drawing may also be applied to the situation in which the System of Systems of one organisation is added to several Systems of Systems of other organisations. Thus from the perspective of this drawing, the entire SLC is parallel and similar if not identical at any level but the lowest.

14.5.2 Mapping the architectural framework into the organization.

If Figure 14-4 is turned on its side, as in Figure 14-5, the traditional hierarchical organisational structure can be seen with the SCCB at the top. Thus from this perspective the System of Systems can be seen to be the same as the traditional system-subsystems configuration which can be mapped into the framework of any organisation of projects. However, when a mapping is made into an existing organisational framework, it will be seen that:

- The terminology used by the organisation and this Chapter will be different.
- The uncoordinated ad-hoc evolution of the System of Systems will become apparent due to the broken, missing links and elements as shown in Figure 14-6. In addition, extra links may become visible. For example, in many organisations the SCCB may not exist and individual CCBs may not have links to one or more external CCBs.

14.6 From the perspective of self-regulating systems

Consider the issue from another viewpoint. An organisation, project or even an element in a sequential process is a system and may be represented as

shown in various ways such as Figure 1-1, Figure 1-3, Figure 4-1 and Figure 14-7. The system turns raw materials (inputs) into products generating waste and profits (wanted and unwanted outputs). There is a control loop which determines what is produced and when[62], based on customer needs. However, what is not usually pointed out is that the control loop embodies a delay. Now consider two systems in series, they may be any sequential processes, such as manufacturing and painting, or subsystem construction and integration. The two subsystems, connected by the production process and the control and status links, may be depicted as shown in Figure 14-8.

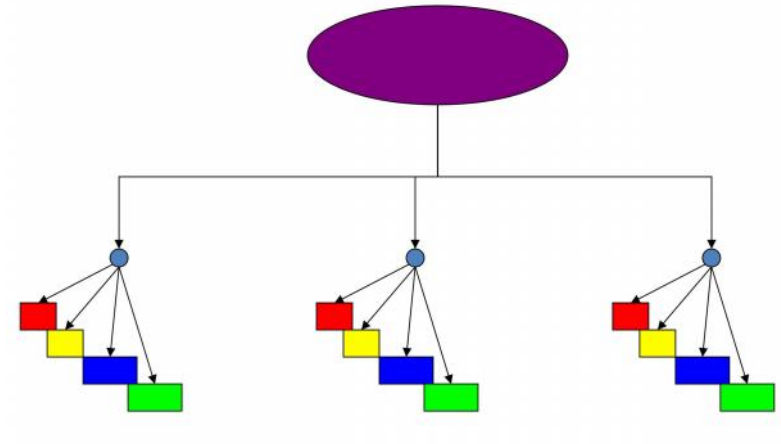


Figure 14-5 Traditional hierarchical organisation

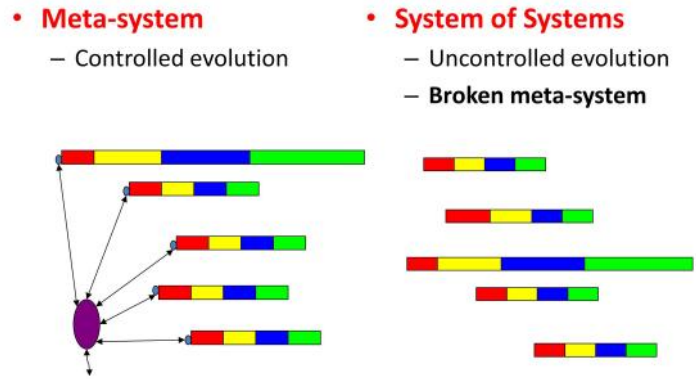


Figure 14-6 Comparison between system of system and Meta-system views

[62] This control loop may be broken in some organisations as a result of poor management.

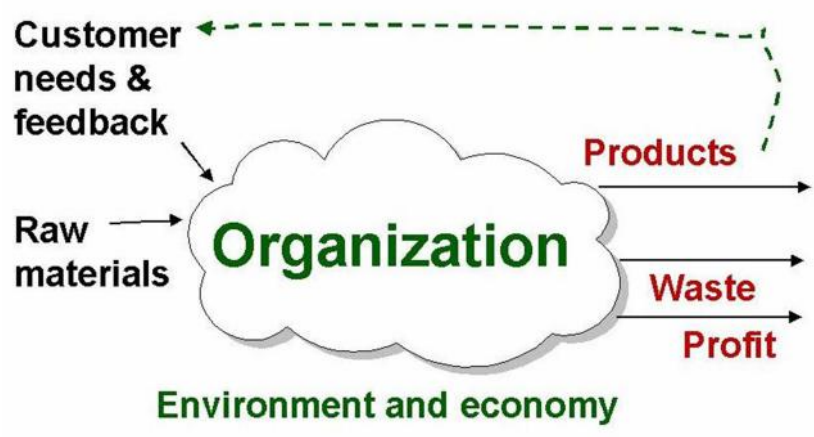


Figure 14-7 The organization as a system

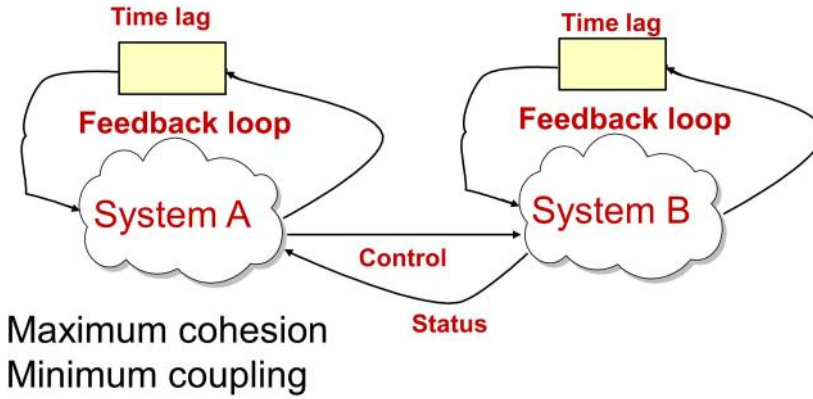


Figure 14-8 Self-regulating systems

If the CCB control is added to the self-regulating subsystems, the arrangement becomes as shown in Figure 14-9. If the drawing is then expanded horizontally along the process dimension and vertically up the control dimension, and the control elements combined to optimise the span of control, the result is the conventional hierarchical organisation structure or system-subsystem chart close to that shown in Figure 4-3, Figure 4-4 and Figure 14-5. The difference being in that in self-regulating systems, decisions are made at the local system level and guidelines are passed down the hierarchy and status information is passed up the hierarchy.

Thus from this perspective as well, if any links are broken, any current individual system acquisition can be considered as part of an uncoordinated ad-hoc evolution of the System of Systems of which it is a part.

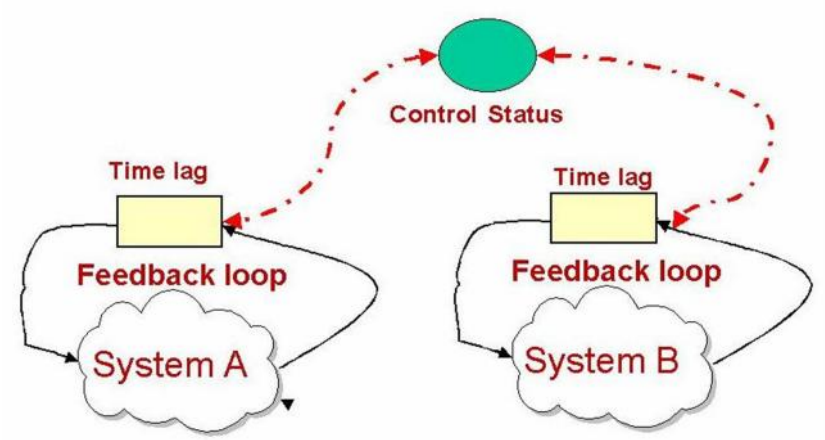


Figure 14-9 Adding control to self-regulating systems

14.7 Gaining control of the system of systems

Since any current individual system acquisition can be considered as part of an uncoordinated System of Systems lifecycle, gaining control of the Systems of Systems is then a matter of:

- baselining the existing System of Systems within a framework for controlled phased evolution as depicted in Figure 14-4,
- performing a gap-analysis to identify the missing elements,
- adding the missing elements to the framework,
- developing the appropriate management tools suitable for both the individual CCBs and the SCCB,
- converting each individual system SLC to the Cataract Methodology,
- developing the transition plan, and
- implementing the plan using the Cataract Methodology.

The transition effort will probably be in converting to the Cataract Methodology and establishing communications paths between the CCBs of the individual systems and establishing the SCCB.

14.8 The suite of tools

The suite of tools needed to control changes are a combination of several existing different and usually unconnected tools such as Requirements Management, Project Management, WBS, Configuration Control, and Cost Estimation, used in the management and engineering paths of the SLC. The QSE for each system and knowledge based tools should provide the capability for a CCB to manage the acquisition of a single system in a cost-effective manner. The recursiveness of the SLC means that the suite of tools should allow

the SCCB to manage the acquisition of the System of Systems in a cost-effective manner. In addition the tools should also work within the IDE (Chapter 3) to provide the capability to perform:

- Impact assessments of the effects of proposed changes before implementation.
- Trade studies on the costs and capabilities of alternative designs at the System of Systems as well as at individual systems levels.

Additional research needs to be carried out into the tools required for the SCCB as well as the degree of abstraction needed to avoid information overload.

14.9 Summary

While managing System of Systems is a complicated problem in an industrial age paradigm, when viewed from an information and knowledge management paradigm, the problem is much less complicated. By integrating Configuration Management across the process and product and using the Cataract Methodology to manage each system within the System of Systems, management of the System of Systems is achievable. However, the necessary information age tools are still evolving out of the current generation of project management, engineering and cost estimation tools.

2002

15 Systems engineering: an alternative management paradigm?

Previous Chapters have examined the organisation, the process it uses and the product it builds. This Chapter looks at the organisation from the perspective of systems engineering and management continuing from Chapter 2. It examines role of systems engineering in the current management paradigm, and revisits the difficulty of establishing a body of knowledge for systems engineering. The Chapter then resolves the difficulty by showing that systems engineering is an alternative management paradigm to the 20th Century Taylor paradigm based on adaptive modifications to "Scientific Management"[63].

15.1 Introduction

We are living in an age of transitions. One of these is the transition from hardware-based systems to software-based systems. While IT underpins the civilization in the early years of the 21st Century, the current systems and software acquisition and development paradigm is characterized by mind-boggling complexity, a turbulent transition from function-based designs to object-oriented designs, and cost and schedule overruns. One reason for this state of affairs is that IT acquisition and development is currently performed in an industrial age management paradigm based on adaptive modifications to the work of F. W. Taylor who system engineered his manufacturing organization to develop the optimal process at the start of the 20th Century. This paradigm is breaking down in the systems acquisitions, IT and software development of the late 20th Century and early 21st Century.

[63] F. W. Taylor system engineered his organization to split the work between management and labour. Management was supposed to plan and design the work, while labour was to implement it in the predetermined one best way.

The process of the conversion of a set of vague, varying and changing user needs into delivered systems and software via a process that can take months or even years is the toughest challenge facing the IT profession in these initial years of the 21st Century. People who can lead the implementation of the IT systems and software acquisition and development process within the cost and schedule constraints are scarce. These people are becoming known as systems engineers and the approach they use is systems engineering. Thus while the world is turning to systems engineering to solve the problems of developing and maintaining the systems underpinning our civilization, systems engineering:

- Is a vague term with many different interpretations. As such, many systems engineers cannot clearly articulate the functions and benefits of systems engineering (Chapter 10).
- Covers a broad spectrum of activities (Chapter 12) and consequently, it has been extremely difficult to establish a SEBoK; something that is critical to the future of systems engineering.

15.2 The Need for a SEBoK

The lack of a SEBoK (Chapter 12) is having a detrimental effect on systems engineering as evidenced by:

- Poorly implemented systems engineering with consequent cost and schedule escalations as well as unnecessary expenditures (Chapter 5). Among the many examples of anecdotal evidence, including the systems engineer with major responsibilities for a shipyard's overseas Combat Systems integration who had absolutely no idea of any of the recognized systems engineering principles (Newland, 1998).
- The continual discussion of fundamental concepts without closure and moving on to the application of the concepts to the real world. A typical example is the discussion on poorly written requirements and their effects, which has not progressed much beyond (Hooks, 1993). The effect of poorly written requirements on costs and schedules needs to be minimized not discussed!
- The continual discussion on the differences between systems engineering and project management. Most of these discussions focus on the technical role of the engineers and the administration and organizing role of the manager. The discussions conveniently ignore the fact that the managers make the decisions and the systems engineers hold the knowledge necessary for making an informed decision. One of the root causes for the breakdown of the Taylor paradigm is that we are now separating the knowledge holders from the decision-makers. The discussion and conflict should terminate with the realization that systems

engineering and project management perform the same functions but in different management paradigms.

- The rediscovery and reinvention of concepts long adopted in overlapping disciplines. As an example, LaPlue et al. discussed the development of a methodology for specifying requirements that describe the behaviour of a system and its interaction with its environment (LaPlue, et al., 1995). In fact they reinvented the environmental and behavioural models of the Ward and Mellor software development methodology (Ward and Mellor, 1985). This situation escalated project costs since the methodology existed and could have been used rather than reinvented.
- The lack of an understanding of fundamental concepts, which is then masked by the excuse that something cannot be done by conventional approaches and a new and complex approach is needed with the requirement for funding the research and the postponement of the delivered implementation.
    - For example the term "System of Systems." Real systems engineers understand the hierarchical concept of meta-systems, systems and sub-systems. And when you understand that concept, you can see that the acquisition of what has become known as a "System of Systems" is just a matter of coordinating an uncoordinated ad-hoc multi-phased time-ordered parallel-processing evolutionary process (Chapter 13).
    - The concept of cost as an independent variable (CAIV), which is a way of complicating just a part of the concept of designing budget tolerant systems using the Cataract approach (Chapter 13). Moreover, to repeat, excessive complexity is a symptom of an underlying problem within the foundation of the current paradigm. Thus complex solutions to any problem are always inherently less than optimal and are the antithesis of the products of true systems engineering.
- Coining new words due to the lack of understanding that words and concepts already exist. System of Systems was discussed above. However, to be fair, systems engineering is not alone in doing this. For example software engineering has recently discovered the technique of "abstracting" or hiding the insides of a component from the designers who integrate the component with other components. Hardware and systems engineers have long been familiar with the concept of a "black box".

15.3 Organization of the SEBoK

If systems engineering is an alternative management paradigm, then the

SEBoK has to be extensive. Chapter 12 proposed that it needs

- Both depth and breadth at the same time.
- To overlap a number of engineering and management disciplines.

Chapter 12 also suggested that a suitable framework for organizing a SEBoK could be based on the Hitchins five-layer model (Section 12.1.2). The broadness of the five layers imply that systems engineers have to be cognizant of what is being published in conferences and journals in several fields including economics, requirements engineering, engineering management, TQM, business, and software engineering as well as systems engineering. The nature of the activities in each of the five layers is such that:

- Systems engineers operating in one layer use a different vocabulary to those operating in another layer, hence, among other manifestations, the multitude of definitions of systems engineering some of which were quoted in Table 12-1.
- The SEBoK could contain almost the entire technical and managerial bodies of knowledge.

15.4 Systems engineering as an alternative paradigm

The major roadblocks hindering the development of a comprehensive SEBoK can be bypassed by the recognition that systems engineering is a different way of doing things with respect to the current paradigm based on adaptive modifications to Scientific Management (Taylor, 1911) namely an alternative management paradigm. Consider systems engineering as a return to an older management paradigm that was used for the construction of projects such as the ancient pyramids, and the 19th Century canals and railroads. Systems, which, within the scope of the tools and technology of their time, presented problems that were just as complex as the problems we face today. This older paradigm can be defined as "a set of activities which control the overall design, development, implementation and integration of a complex set of interacting components or systems to meet the needs of all the users". This sentence just happens to be a DERA definition of systems engineering (DERA, 1997) quoted in Table 12-1.

Kuhn writes that an alternative paradigm is a reconstruction of the field from new fundamentals, a reconstruction that changes some of the field's most elementary theoretical generalizations as well as many of its paradigm methods and applications (Kuhn, 1970). The field under discussion in this Chapter is "acquiring and delivering systems and software that meet the changing needs of the user on time and within budget". If systems engineering is a different management paradigm for this field, then to meet Kuhn's requirement for an alternative paradigm it has to

- Resolve conflicts that cannot be readily resolved within the current paradigm.
- Incorporate management and other activities that are currently performed by non-systems engineers who may be competing (or overlapping) with systems engineers.

Systems engineering seems to do both. Some examples (in chronological order of publication) are

- Eisner lists a general set of 28 tasks and activities that is normally performed within the overall context of large-scale systems engineering (Eisner, 1988). The full range of activities is commonly called 'specialty skills' because some people spend their careers working in these specialties. Thus according to Eisner systems engineering overlaps at least 28 engineering specialties.
- The DSMC definition of systems engineering quoted in Table 12-1 is "The management function which controls the total system development effort for the purpose of achieving an optimum balance of all system elements. It is a process which transforms an operational need into a description of system parameters and integrates those parameters to optimise the overall system effectiveness" (DSMC, 1996). Notice the use of the term "management function"!
- As mentioned in Chapter 12, Lewis provides case studies in software IV&V (Lewis, 1992). Yet the words "IV&V engineers" in those case studies could be replaced by the words "systems engineers" and the cases would be just as appropriate in a book on systems engineering instead of a book on IV&V.
- Deming laments the failure to manage the organization as a system (Deming, 1993) page 30). He advocates replacing Management by [individual component] objectives (MBO) by studying the theory of a system and managing the components [setting the objectives] for optimisation of the aim of the system. He further states, "*We are living under the tyranny of the prevailing style of management. Most people imagine that this style of management has always existed, and is a fixture. Actually, it is a modern invention, a trap that has led us into decline*" (Deming, 1993) page 50).
- Deevy writes that management consulting has become a major growth industry with American companies now spending over 7 Billion [dollars US] each year on outside advice (Deevy, 1995) page 25). Thus Deevy seems to imply that engineers are expected to know how to engineer, physicians are expected to be competent practitioners of medicine, yet the current management paradigm has reached the stage where it does not expect managers to know how to manage!

- Roe in discussing the role of the systems engineer and project manager states that the knowledge and skills of system engineering are the same as those of project management in the areas of management expertise, technical breadth and technical depth (Roe, 1995). The difference in application, according to Roe, is that the system engineer has more technical breadth, while the project manager has more management expertise. Roe concludes with "*to perform effectively, the project manager must be a system advocate. He must learn the multidisciplinary approach and embrace the systems engineering methodology. The project manager and the systems engineers share common objectives, i.e., to plan, design, and deliver a system that meets the customer's needs. Thus, they must avoid conflict and work cooperatively to attain the project's goals*."
- Roe also writes, "*Integrated Product Development (IPD) is becoming a fact of life for companies doing business with the government, and for those who have found that it provides an edge in the modern global market. Success stories abound. But it has not come easy to any organization. Each has found it necessary to make fundamental changes and to convince upper management that the rewards of undergoing the cultural shock of reengineering the organization will more than justify the pain. Lessons have been learned and are still being learned and applied to readjust the process. New management techniques, coupled with aggressive application of systems engineering, will prove to be key elements of IPD*" (Roe, 1995).
- Martin Cobb stated "*We know why projects fail; we know how to prevent their failure --so why do they still fail?*" This statement has become known as Cobb's Paradox (Chapter 6).
- Sheard described twelve roles of the systems engineer that are occasionally or frequently assumed to constitute the practice of systems engineering (Sheard, 1996)[64]. According to her, some of the roles fit naturally as [system development] lifecycle roles; others fit the program management set of roles, while still others are not normally thought of in either group.
- Chapter 2 analyses the functions performed by systems engineers and shows that there seems to be no unique body of knowledge to systems engineering. It states that all of the activity performed by systems engineers, apart from possibly requirements and interfaces, are also performed by other types of engineers.
- Watts and Mar write that systems engineering and project management should be integrated (Watts and Mar, 1997). They state that in many

[64] See Section 23.1.

project environments, system engineering and project management are managed separately. This situation is aggravated by the discipline segregation by universities and by the corresponding professional organizations. As a result of this separateness, project managers futilely try to manage cost and schedule without managing technical content while the technical providers, ambivalent to the cost and schedule consequences, pursue superior technical solutions.

- Eisner expands his earlier list (Eisner, 1988) and discusses 30 tasks that form the central core of systems engineering (Eisner, 1997) page 156). The whole area of systems engineering management is covered in just one of the tasks. Eisner states that "*not only must a Chief Systems Engineer understand all 30 tasks; he or she must also understand the relationships between them, which is an enormously challenging undertaking that requires both a broad and deep commitment to this discipline as well as the supporting knowledge base*".
- Drucker wrote, "*Full realisation of the systems concept in manufacturing is years away. It may not require a new Henry Ford. But it will certainly require very different management and very different managers*" (Drucker, 1993) page 315).
- Bottomly et al. studied the roles of the systems engineer and the project manager and identified 185 activities and their competencies (experience and knowledge) (Bottomly, et al., 1998). Their findings included:
    - No competency was assessed as being purely the province of systems engineering.
    - There is no sharp division between the two disciplines (systems engineer and the project manager) even at the level of individuals.
- Hitchins develops four litmus tests for systems engineering. He states that the conclusion to be drawn from the application of the litmus tests to some everyday industrial practices is that many practices which present themselves as systems engineering are either inappropriately, titled, or are, perhaps, only part of a much richer story. Even classic systems engineering is seen to have some limitations in practice (Hitchins, 1998). Hitchins' penultimate sentence states, *"[systems engineering] is a philosophy and a way of life*".
- Brooks and Mawby recognize systems engineering and project management as separate disciplines, both of which regard decision making as their territory (Brooks and Mawby, 1998). They add that conflicts of this type are often reinforced by a traditional functional organization creating barriers between management and engineering so that any projects conducted in that environment will benefit from the full potential of an integrated approach to systems engineering and project management.

- Robson discusses the paradox in which the tribes [of the various work disciplines] need to adopt a systems engineering approach, and, "*the challenge is to achieve this without the provocation of the 'systems engineers are taking over the world'!"* (Robson, 2001).

The overlap in, and the difficulty of allocating, the roles of systems engineers, can be overcome in the manner of overcoming any paradox, namely by a change of perspective or paradigm.

In conventional systems engineering terminology, there is a system (an organization) that no longer can provide the capability to meet the needs of its customers. Its evolutionary changes seem to have reached the point of diminishing returns and perhaps way beyond. Adding layers of complexity is not the solution. It is time for a replacement system. The first stage in the process of replacing the system is to recognize the need for a replacement. Chapter 2 is based on the first paper (within the systems engineering community) that seemed to recognize the situation and questioned the role of systems engineering within the current management paradigm stating

"Systems engineering is a discipline created to compensate for the lack of strategic technical knowledge and experience by middle and project managers in organizations functioning according to Taylor's 'Principles of Scientific Management.

Most of today's systems engineers really appear (work as) to be Requirements and Interface Engineers. They have the responsibility to validate the requirements since there's little point in building a system which conforms to requirements if the requirements are incorrect. Perhaps those are two missing 'ilities' in the current paradigm.

Project management, BPR, concurrent engineering, TQM and theoretical systems engineering all seem to be attributes of the same function; namely producing a product to (the correct) specifications by an organization within the constraints of resources, budget and schedule. Remember (MIL-STD-499A, 1974) was written for systems engineering management and (MIL-STD-499B, 1992) changed the focus to systems analysis and control. This overlap or duplication seems to be due to defects in the current organizational structure, and in the case of systems engineering, the transition in technology from hardware to software. We need a new organizational paradigm to simplify the organization such as the one proposed in Chapter 3 and within that paradigm, there still is a need for someone to have a strategic perspective of the entire system."

Transition is taking place. John W. Campbell Jr. wrote that changes which change the rules of the game and require people to shift their perspective are resisted. It is an emotional issue and cannot be settled by logic (Campbell, 1960). Kuhn wrote that changes which require people to unlearn what they already know is correct are resisted very strongly (Kuhn, 1970).

Consequently, it is nigh impossible for managers to unlearn what they know is the correct way to manage. So the failure of the process improvement initiatives which require managers to unlearn what they know is the correct way to manage (paradigm), based on their years of experience on the job, does not come as a surprise to anyone familiar with Kuhn's work. Yet the recent books on improving management all state that management should unlearn something they do now and change to a different way of doing things. Unwillingness to unlearn is a major cause of resistance to change. No wonder that Drucker stated that a paradigm shift takes about 25 years, namely the time it takes for the unwilling to unlearn proponents of the old paradigm to retire (Drucker, 1985).

The systems organization is beginning to appear. For example, Crosby stated that organizations have recognized the need for a drastic change in their way of doing things (Crosby, 1992) page 10). Some organizations have almost made the paradigm shift. As companies struggle to boost efficiency and become more market responsive, they have stripped out layers of management, broadened the remaining span of control and employed technology. Up front planning ensures Quality. The trend in enlightened management is to delegate the decision-making and control to the employees doing the actual work. Employees are taught how to analyse and solve Quality problems with minimal management supervision, and in some instances, the lines between workers and middle management blur and even vanish (Kanter, 1987) page 60). Organizations still use the term "management", yet the functions performed by these managers map into the definition of the functions performed by systems engineers. Surely the next logical step must be to eliminate managers completely, thereby solving Cobb's paradox, and adopt the systems engineering paradigm of management. As Drucker wrote "*the form which management will take may be quite different tomorrow. The restraints, the controls, the structure, the power, and the rhetoric of management may change drastically*" (Drucker, 1980) page 226), which means changing their corporate culture.

Thus organizations will suffer upheavals in making the paradigm shift. Not every organization is going to be able to make the paradigm shift. Many will fade away like the 381 companies that dropped out of the Fortune 500 listings between 1955 and 1990.

Management as a practice is very old going back to the beginnings of recorded history (George, 1972; Drucker, 1995). Today's hierarchical management structures evolved to control the railroads and mass production factories of the industrial age, the late 19th and early 20th centuries, and have largely become redundant because information technology can now route and filter organisation from the corporate office to the production line faster (Davidow and Malone, 1992) page 163). However the discipline of modern management is only about 60 years old, emerging in the post-war US (Drucker, 1995) page 250);(Johnson, 1997). Thus management and systems

engineering emerged as disciplines at about the same time and have been concerned with the same overlapping functions in organisations. Middle management may also be an artefact due to the lack of information technology in the industrial age in the evolution of society from agrarian to information.

15.5 Conclusions

The conclusions of this Chapter are:

- The lack of a SEBoK is detrimental to systems engineering, consequently one must be established quickly.
- However, a comprehensive SEBoK covering all five layers of systems engineering many never be established because it would have to contain almost the entire technical and managerial bodies of knowledge for hard systems as well as the body of knowledge relating to soft systems.
- The difficulties in placing the role of the systems engineer within the current management paradigm and the scope of the SEBoK seem to confirm the view that systems engineering is a different management paradigm to that of the Taylor based organizational paradigm.
- The world is in the process of a transition from the Taylor based organizational paradigm to the systems engineering paradigm of management.
- Managers are artefacts of the lack of information technology in the industrial age in the evolution of society from agrarian to information.

2002

16 Does object-oriented system engineering eliminate the need for requirements?

This Chapter uses the product viewpoint to examine system engineering and object-oriented methodologies and then shows both that systems engineering is inherently object-oriented and that object-oriented languages such as the UML may be used to document the user's needs in a manner that can be used by developers. The Chapter also suggests a next generation tool concept that can be used to hold both user and developer representation of the user's needs as an alternative to, and an improvement on, text mode "requirements", reducing communications problems and hence increasing the reliability of the shared meaning of the user's needs amongst all stakeholders.

Kasser and Schermerhorn wrote that systems engineers needed to use a methodology that seamlessly interfaced to the software development methodology to avoid delays and errors in translation from the system requirements to the design phase (Kasser and Schermerhorn, 1994b). So as software engineering has adopted an object-oriented perspective, system engineering has attempted to do the same. However, at the same time, software engineering seems to be discovering for itself, the advantages of using the functional decomposition approach coupled with an object-oriented perspective (Chang and Hua, 1994). This Chapter presents a brief analysis of both systems engineering and software engineering, shows that systems engineering is inherently object-oriented, and then suggests ways that three elements of the object-oriented software engineering paradigm may be used to enhance systems engineering. Furthermore, when these elements are coupled with a 'yet to be built' next generation requirements tool text-mode based requirements may become an academic issue.

Systems engineering has traditionally been viewed as following a process known as the SDLC to bring a system into existence. Systems engineers are the people who implement the systems engineering process. Blanchard

and Fabrycky provide the following definitions (Blanchard and Fabrycky, 1981).

- **System** - an assemblage or combination of components or parts forming a complex or unitary whole.
- **Elements of a system** - Systems are composed of
    - **Components** - the operating parts of a system consisting of input, process, and output. Each system component may assume a variety of values to describe a system state as set by control action and one or more restrictions.
    - **Attributes** – the properties or discernible manifestations of the components of a system. These attributes characterize the parameters of a system.
    - **Relationships** – the links between components and attributes.
- **Function** – the purposeful action performed by the system.
- **Systems and subsystems** – Every system is made up of components, which can be broken down into smaller components. If two hierarchical levels are involved in a given system, the lower is conveniently called a subsystem.

Blanchard and Fabrycky also state that systems engineering *per se* is not considered as being an engineering discipline in the same context as the technical specialties it represents (Blanchard and Fabrycky, 1981). The need for systems engineering is present because many of the engineering specialists in one or more of the conventional engineering areas do not have the experience or knowledge to consider the system as a whole as well as knowledge of the non-functional technical specialties. Now the goal of systems engineering is to provide a system that (Section 9.2):

- Meets the customer's requirements as stated when the project starts.
- Meets the customer's requirements as they exist when the project is delivered.
- Is flexible enough to allow cost effective modifications as the customer's requirements continue to evolve during the operations and maintenance phase of the system lifecycle.

In the traditional systems engineering paradigm, the systems engineers partition the system by function such that each component of the system provides some of the desired capability. Or in object-oriented terminology, they encapsulate the components of the system by localization of functions to produce a system. Moreover, Hambleton points out that system engineers do not always map functions to physical components in a 1:1 mapping. In the example of a bicycle, the front wheel (physical) can be considered as an element in at least two virtual systems (steering and drive) (Hambleton,

1999). In actual fact, a system may be represented by many kinds of objects, appropriate to the analysis in process.

In practice, systems engineering tends to focus on the "functional capabilities required" to build a physical system (Eisner, 1988). The "non-functional capabilities" needed or "non-functional requirements" tend to be shunted aside during development causing later problems when the system "as-delivered" does not perform its intended task in its operating environment. This is ironic, because the physical "as-delivered" system has both functional and non-functional properties.

16.1 The object-oriented paradigm

While software engineering claims to have invented the object-oriented paradigm, it was documented as an analysis methodology at least as early as the 12th Century. An object is characterized by its (Maimonides, circa 1200) pages 69-70):

- definition;
- part of its definition, namely what it inherits from a parent;
- attributes;
- relationships with other objects;
- internal actions.

Van Vliet states that there are different viewpoints of what the notion of an object entails, and provides the following definitions (Van Vliet, 2000):

- **The modelling (European) viewpoint –** an object is a conceptual model of some part of a real or imaginary world. Each object has an identity that distinguishes it from all others. Objects have substance: properties that can be discovered by investigating the object.
- **The philosophical viewpoint** - objects are existential abstractions fall into two categories
    - ephemeral - exist for a period of time, and
    - eternal - live forever.
- **The software engineering viewpoint** - objects are data abstractions, encapsulating data as well as operations on the data. This viewpoint stresses locality of information and representation independence.
- **The implementation viewpoint** - a contiguous structure in memory (software); may be composite or aggregate (possesses other objects)
- **The formal viewpoint –** a state machine with a finite set of values, and a finite set of state functions.
- **The problem domain viewpoint** - objects are abstractions reflecting the capabilities of a system to keep information about it, interact with it, or both.

- **The solution viewpoint** - objects are encapsulations of attribute values and their exclusive services.

Van Vliet also notes that objects with the same set of attributes are in the same class. Individual objects in a class are called instances of the class. Objects encapsulate state and behaviour where behaviour is described in terms of services provided by the object, and services are invoked upon receipt of a message from another object. The behaviour of an object is described by systems engineers in terms of the functions performed by the object, while the software engineer describes the same behaviour in the terminology of "services provided by the object." Apart from the semantic difference in the use of the words "functions" and "services", Van Vliet's objects are systems and subsystems in systems engineering terminology.

The object-oriented paradigm, based on Structured Programming concepts, embodies its own terminology in three basic principles:

- **Encapsulation** - the concept of placing data and processes, or methods or routines that operate on that data together and combining them to create a structure (object) that contains both. In systems engineering terminology encapsulation is design, and an object would thus seem to be a subsystem.
- **Inheritance** - the concept that new objects are derived from existing objects. These new objects can add to or vary their behaviour with respect to the behaviour of the parent objet. This is the main feature that can lead to re-use of existing software. Inheritance can be both physical as well as logical showing up in the hardware world as well. For example, a new communications component will inherit characteristics from existing components.
- **Polymorphism** – the concept that causes different types of objects derived from the same parent object to be able to behave differently when instructed to perform a same-named method in a different implementation.

Van Vliet classifies object-oriented design as a middle-out design method. The set of objects identified during the first modelling stages constitutes the middle level of the system. In order to implement these domain-specific entities, lower level objects are used. This concept is no different to the systems engineer designing a system using subsystems. The only difference may be in the scope of the design activity. Hardware engineers designing a printed circuit board (in general) do not design their own customized integrated circuit components but use COTS ones from a manufacturer's range. Today's software engineer, working with components may have to design and construct their own components to provide capability not available in COTS components in order to build the software system.

16.2 Different perspectives of systems

Consider the process-product production sequence in the up-front activities of object-oriented systems engineering and object-oriented software engineering as shown in Figure 16-1. Object-oriented software engineering generates objects directly from the Use Cases, while current object-oriented systems engineering first develops requirements and then develops objects. Electrical engineers will recognize that Figure 16-1 shows the object-oriented software engineering approach short-circuiting the requirements stage of object-oriented systems engineering. However, object-oriented systems and software engineering do have different perspectives of systems as discussed below.

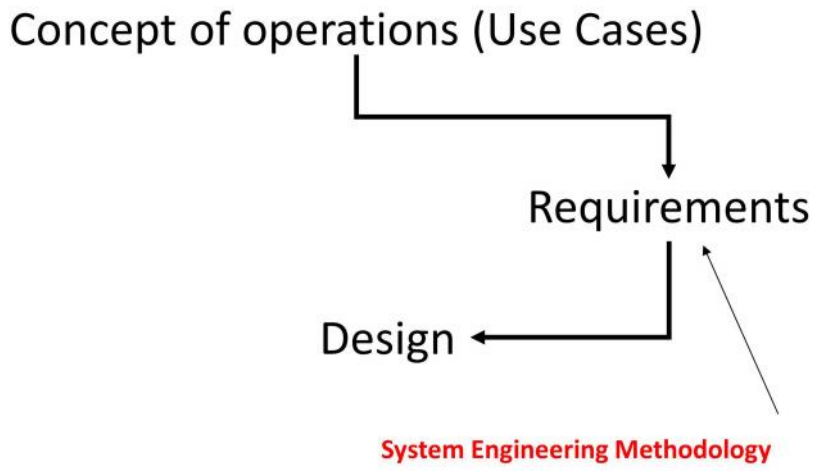


Figure 16-1 Process-product production sequences

16.2.1 Software engineering

Software engineering considers information flows, and functional requirements. Non-functional requirements tend not to be considered in the software design. In its early days, software engineering tended to ignore the physical domain, since the software operation was on a single hardware platform, however with the advent of distributed systems and client server techniques, software operates across several hardware platforms and the system needs to be optimised as a whole.

According to Van Vliet, the object-oriented approach to systems analysis and design involves the following three major steps in an iterative manner.

- Identify the objects.
- Determine their attributes and services. The functions performed by the system are represented by the services performed by the objects.

- Determine the relationships between the objects.

16.2.2 Systems engineering

In addition to the information flows, and functional requirements, systems engineering also has to consider the physical properties of a system and the non-functional requirements. Systems engineering employs different models or perspectives for different purposes. For example, Hatley and Pirbhai's methodology employs three models (Requirements, Architecture and Specification) (Hately and Pirbhai, 1987), Menzes et al. discuss viewpoint-based requirements engineering, and its advantages (Menzes, et al., 1999) while the DODAF calls out 26 different views (DoDAF, 2004). In systems engineering, each model abstracts attributes so that only the aspects being studied are seen or in object-oriented parlance uses the concept of "information hiding." For example, when considering the characteristics of a laptop computer one model may consider the physical aspects of cabinet, keyboard, disk drives, etc.; a second may consider the underlying hardware architecture of the computer; and yet a third may examine aspects of heat transfer inside the cabinet. Systems engineering may use different objects in each model. Lagakos et al. write that "*The object-model formulation views a system as a group of interacting objects that work together to accomplish system objectives and satisfy system requirements. Use-case and domain models provide a visual representation for high-level system functionality and system design"* (Lagakos, et al., 2001). If they had used the word "interacting sub-systems" instead of "interacting objects" they would have reiterated one of the maxims of systems engineering in the language of traditional systems engineering.

The traditional functional analysis and allocation can be described as following two major steps in an iterative manner.

1. Identify the functions performed by the system.
2. Partition the system such that each combination of one or more functions is performed by a physical subsystem.

16.2.3 Similarity in analysis and design

The same two factors are implicit in both systems engineering and object-oriented software engineering, namely

1. The designer determines that the relationships between the objects/subsystems are such that sum of the capability of the subsystems and the desired emergent properties of the system when constructed meet the needs of the customer.
2. The system is to be designed with minimal coupling between subsystems/objects and maximum cohesion within a subsystem/object, namely good engineering practice.

Thus systems engineering is inherently an object-oriented approach; however, the perspectives of systems engineering and object-oriented software are different. Somerville states that the initial stage of function-oriented design relies on identifying functions which transform their inputs to create outputs while the system is treated as a black box (Sommerville, 1998). The object-oriented design focuses on the entities within the system and considers the functions as part of the entities. Each design approach will generally produce different system decompositions. Somerville states that the most appropriate design strategy is often a heterogeneous one in which both the functional and object-oriented approaches are used. This is not surprising as systems engineering uses the identical approach in a 'design to inventory' situation to assemble a system out of components in the inventory.

Chang et al. propose a concept known as function-class decomposition (FCD) (Chang, et al., 2001). FCD, an iterative process, applies a top-down approach to decomposing a system while simultaneously identifying and grouping classes into functional modules, with each module representing a specific functionality which the system requires. Chang et al. seem to have discovered and applied the systems engineering process!

16.3 Enhancing systems engineering

In summary, while the object-oriented approach is little different from traditional systems engineering, it explicitly emphasizes three elements that may readily be used to enhance systems engineering, namely:

- **Properties** which have values and attributes. Thus capability and requirements may be expressed as properties provided or properties desired.
- **Inheritance** which explicitly moves experience from designer into design. Whereas in systems engineering, the non-functional aspects of a system were often a function of the experience and expertise of the designer, the concept of inheritance can be used to inherit (non-functional and other domain) properties and eliminate some of the "missing" requirements of today'' paradigm.
- **Encapsulation or design** which can be non-functional (i.e. data, and process) as well as the traditional functional approach to partitioning a system.

16.4 The (process) interface between systems and software engineering

The interface has traditionally been at either the SRR or the System Design Review (SDR). System engineers have focused on generating requirements

to ensure that the as-built system is fit for its intended purpose. However, the current paradigm produces poorly written requirements (Hooks, 1993; Tran and Kasser, 2005) and various approaches have been proposed since then to alleviate the situation without much success. For example:

- Jacobs states that a 1997 analysis of the software development process performed at Ericsson identified "*missing understanding of customer needs*" as the main obstacle for decreasing fault density and lead-time (Jacobs, 1999). Related findings were aggregated under the heading "no common understanding of 'what to do'". The counter measures to overcome these problems focused on testing the quality of the requirements rather than producing good requirements. There was no proposal on how to get clear requirements, nor was there a clear understanding of what a clear requirement was.
- Goldsmith states that the process of "*defining business requirements is the most important and poorest performed part of system development*" (Goldsmith, 2004).

Thus from this viewpoint, in practice requirements do not seem to be a useful communications tool for translating between user needs and the system and software designers.

16.5 Is there an alternative to "requirements"

Systems engineering is focused on dealing with well-structured problems (Jackson and Keys, 1984). However, the problem of poor requirements is complicated and ill-structured, and hence not solvable by the traditional systems engineering process. As "requirements" are still poorly implemented after all these years, perhaps they should be eliminated or bypassed (automated). Kasser proposed a tool to improve the wording of requirements[65], but a greater degree of improvement should be achievable by replacing written requirements (Kasser, 2002c). Consider ways in which this might occur.

Gabb et al. define a requirement as "*an expression of a perceived need that something be accomplished or realized*" (Gabb, et al., 2001). Van Gaasbeek and Martin quote Dahlberg as stating that "*we don't perform systems engineering to get requirements*" (Van Gaasbeek and Martin, 2001) and add "*we perform systems engineering to get systems that meet specific needs and expectations*." The focus is on user needs, not requirements. What systems engineering appears to have forgotten is that requirements are used to document user needs in a verifiable manner, Requirements are a means, not an end. There is nothing divine about requirements; they are

[65] The tool, FRED, evolved into Tiger Pro shown in Figure 17-3.

just a convenient poorly-used tool for translating customers' needs into a system that should be built.

Requirements are developed as an intermediate work product in the system development process, and are developed to provide formal communication between the stakeholders. Writing text-based unambiguous requirements for combinatorial and sequential scenarios in the form of imperative construct statements is difficult. Timing and state diagrams are often used within the context of the SRD to provide the necessary information. Thus the concept of stating user needs (under certain circumstances) via diagrams is already in use in systems engineering. Sutcliff et al. proposed reducing human error in generating requirements by analysing requirements using an approach of creating scenarios as threads of behaviour through a Use Case, and adopting an object-oriented approach (Sutcliffe, et al., 1999).

16.6 The UML Perspective

The UML (UML, 1999; 2005) being a language has no inherent limitation on the number and types of objects, and is extendable. There thus seems to be no reason why UML should not be used to represent viewpoints or models at the system level as well as the software level. For example Holt applies UML to systems design significantly predating the development of SYSML[66] (Holt, 2001), the modification to UML for systems engineering. Holt provides practical examples that show how the UML can be applied to non-software-based systems. The UML perspective on complex systems can be summarized as:

- Best approached through a small set of nearly independent views.
- No single view is sufficient.
- Every model may be expressed at different levels of fidelity.
- The best models are connected to reality.

This perspective is little different to that of Checkland (Checkland, 1991) and Kline (Kline, 1995). The UML is a modelling language for specifying, constructing, visualizing, and documenting, the artefacts of a software-intensive system. The UML is not a process nor is it a methodology. This fact does not seem to be appreciated in the systems engineering community, as for example, Gibbons writes "*The UML has provided a methodology that encompasses many of the up-front systems engineering activities in an otherwise object-oriented-based program*" (Gibbons, 2001). UML can be used to document systems engineering products in those up-front systems engineering activities within conventional systems engineering methodologies. UML diagrams for models cover:

[66] SYSML can be thought of as providing a standard set of extensions.

- Use cases;
- Classes;
- Behaviour in terms of state, activity, and interaction;
- Charts - showing sequence and collaboration;
- Implementation, aspects of components and deployment.

UML has a four-layered architecture, which can result in systems being constructed from the centre outward. The focus is on Use Cases, which drive the design. This is little different to the systems engineering approach in which the OCD is one of the two critical documents in the SDLC (Kasser and Schermerhorn, 1994b). Gabb summarizes the purpose of the OCD as describing the operation of a system in the terminology of its users stating that it may include identification and discussion of the following (Gabb, 2001):

- Why the system is needed and an overview of the system itself.
- The full SLC from deployment through disposal.
- Different aspects of system use including operations, maintenance, support and disposal.
- The different classes of user, including operators, maintainers, supporters, and their skills and limitations.
- Other important stakeholders in the system.
- The environments in which the system is used and supported.
- The boundaries of the system and its interfaces and relationships with other systems and its environments.
- When the system will be used, and under what circumstances.
- How and how well the needed capability is currently being met (typically by existing systems).
- How the system will be used, including operations, maintenance and support.

Gabb also provides the traditional systems engineering perspective when he writes that "*an OCD is not a specification or a statement of requirement - it is an expression of how the proposed system will or might be used, and factors which affect that use. As such it is not obliged to follow the 'rules' of specification writing and can be relatively free in its language and format. Generally it will contain no 'shalls'*". However, there is nothing to preclude the use of Use Cases for describing all of Gabb's points in a verifiable manner in a manner understandable to both users and developers.

Lagakos et al. state that a Use Case is simply a set of system scenarios tied together by a common user goal (i.e., aspect of system functionality), and describes a way in which a real-world actor would interact with the system (Lagakos, et al., 2001). According to them, a Use Case specification contains:

- A list of actors (actors are anything that interfaces with the system externally);
- A boundary separating the system from its external environment;
- A description of information flows between the actors and individual Use Cases;
- A description of normal flow of events for the Use Case, and
- A description of alternative and/or exceptional flows.

Gabbar et al. state that UML has been proven to be an efficient and comprehensive approach that can describe all three dimensions of the physical aspects of a production plant (static, behaviour and function) (Gabbar, et al., 2001). However, Jorgensen writes, "*Use Cases do not replace requirements*" (Jorgensen, 2001) and Ogren points out that the UML is vague when it comes to requirements capture, and there is no precise definition of what a Use Case should look like (Ogren, 2001). In systems engineering terminology, the UML states the requirements for a Use Case, but does not provide a design. Repeating the statement made above, the UML is a language not a methodology. As such while it may be used for documenting requirements, it does not have an inherent methodology for capturing them.

16.7 Replacing "requirements" by properties

So, if Use Cases can represent the user's needs in a manner verifiable by all stakeholders, then an improvement on the current text-mode based requirements paradigm will have been made. The use of Use Cases driving an object-oriented approach describing properties of components can provide the same representation of user needs as that of "requirements" if each property consists of an attribute and a value.

Consider "property" as the totality of the attribute and its value (functionality and Quality criteria). Then requirements can be stated as the properties needed, and capability can be stated as the properties measured or exhibited by the object. The words functional and non-functional requirements no longer have to be used. When the system is broken down into subsystems each property (attribute and value) is allocated subsystem elements[67]. Traceability of properties (functional and non-functional) is built into the approach.

The objective perspective shown in Figure 16-2 lists the properties of a component[68]. Each property has an attribute, which has some value. In the example of a communications object, as far as performance attributes are

[67] Desired emergent properties can be allocated to a virtual subsystem.

[68] This is a typical Delphi display of the properties of a software component being used to display the properties of a subsystem.

concerned, the data input attribute has a value of 1000±10 units, the Data output attribute a value of 1000±10 units, etc. The services (functions) performed by the object have to do with ingesting the data input, performing some action on the data and then forwarding processed data. A few of the non-functional attributes such as reliability (Mean Time between Failures (MTBF), and Mean Time to Repair (MTTR)) and operating temperatures are also shown.

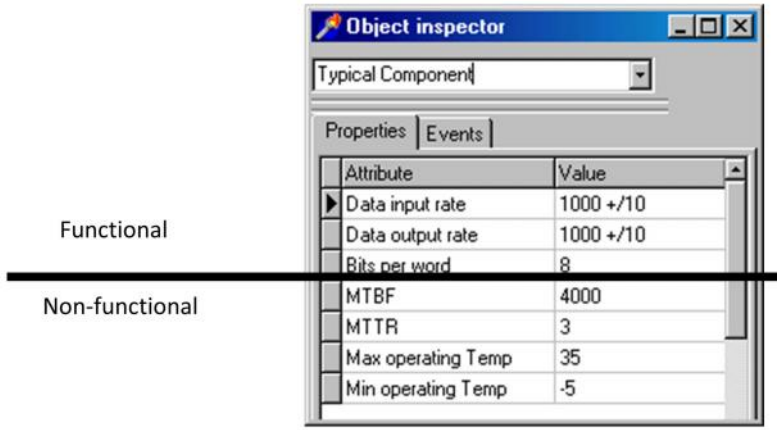


Figure 16-2 The object view

16.8 The next generation of requirements tools

From a historical perspective, user needs, expressed as "requirements" were originally transmitted in text-mode requirements documents and specifications. As tool technology advanced requirements databases were employed and these documents became printouts or reports from the databases. Today's requirements tools tend to operate at this level. Systems and software engineers using Use Cases can employ tools using the appropriate representation provided by the tools at hand and a new tool.

Visualize a project management tool. The project management tool stores information and presents several views (abstractions) to the user. Typical views are GANTT and PERT charts. In a similar manner, a new type of object-oriented systems engineering tool could present information in the user diagrammatic format (e.g. process flow charts) as well as in developer format (UML or whatever development language becomes the prevalent paradigm in the future).

Thus the next generation of network centric systems engineering tools connect via the corporate IDE may have the capability to describe user needs in ways that are both object-oriented and better than the current generation

of text-mode based requirements tools. In addition, once an object-oriented systems engineering/object-oriented software engineering language becomes the language of design, other currently troublesome issues may become moot. For example, the design of interfaces between different engineering tools will be easier. Interface designers will just face the relatively simple problem of transferring the content of the language rather than the thorny problem of "sharing meaning" (Harris, 2000).

16.9 True object-oriented system engineering

Systems engineering is inherently object-oriented and needs only a slight enhancement to seamlessly interface with function-class decomposition based object-oriented software engineering. This seamless interface must be implemented by simplifying the current process, not by adding a layer of complexity between systems and software engineering. The proposal for simplification herein is to make a few changes to the way systems engineering is performed, namely:

- Use the concept of inheritance so that requirements" are not missed by design engineers lacking experience and domain expertise.
- Partition the system for optimal minimally coupled system level component designs. This will mean that system and software engineers will have to work together in the up-front stages of the SDLC before the partitioning into subsystems takes place. This no different to the system engineer working with other technical engineering specialists such as reliability, manufacturing and thermal engineers.
- Stop referring to functional and non-functional requirements and start referring to properties needed (by customers) and properties provided (by components in existence or to be built).

16.10 Conclusions

Systems engineering is indeed inherently object-oriented. The major differences in the object-oriented approach between systems engineering and software engineering seem to be that software engineering tends to pick one model or solution early in the design process and run with it as mentioned in Chapter 12 while textbook systems engineering looks at alternatives and then selects the optimal one based on a set of evaluation criteria. Thus systems engineering seems to operate at the UML meta-model level.

Using an object-oriented approach and the appropriate next generation tools to capture and track properties should result in more systems being built correctly the first time avoiding costly rework after delivery.

Systems engineering is all about ensuring that all the properties of the system as delivered (system capability) are at least equal to the properties of the system needed (system needed). Thus it is requirements free.

2003

17 Object-oriented requirements engineering and management

This Chapter takes a process-product object-oriented perspective on requirements engineering. The Object-oriented requirements engineering described in this Chapter is an approach to encapsulating information about the process and product, as well as functionality into a requirements object. This Chapter:

- Identifies candidate properties of a requirement object based on information in the process (development, management and test and development streams of work in the SLC) as well as information about the product needed.
- Describes some of the functionality that could be added to the requirements object.
- Concludes that object-oriented requirements engineering and management can effect a significant reduction of the problems currently encountered in the SLC due to poor requirements engineering and management.

The systems and software development industry is characterized by a paradigm of project failure and cost and schedule overruns. The situation has been described by Cobb's Paradox (VOYAGES, 1996), which stated "*We know why projects fail, we know how to prevent their failure --so why do they still fail?"* One of the known contributing causes of these project failures is poor requirements engineering and management, which has been repeatedly and widely discussed and documented for at least 10 years (Hooks, 1993; Kasser and Schermerhorn, 1994a; Jacobs, 1999; Carson, 2001) etc. However, this continual documentation and discussion of the problem of poor requirements engineering and management has not resulted in a practical solution to the problem. This Chapter contains a preliminary introduction to an object-oriented approach to requirements engineering that might help to reduce the contribution of poor requirements engineering and management to project failures.

17.1 Requirements engineering and management

As quoted in Section 8.3 requirements engineering and management are evolving as evidenced by changes in the definition of the term. There has since been recent recognition that a requirement is more than just the imperative statement. For example, both Alexander and Stevens and Hull et al. discuss additional properties of the text-based requirement (e.g. priority and traceability) in conjunction with improving the writing of requirements (Alexander and Stevens, 2002; Hull, et al., 2002). Now the IEEE States (IEEE CCCC, 2003), "*Requirements identify the purpose of a system and the contexts in which it will be used. Requirements act as the bridge between the real world needs of users, customers and other stakeholders affected by the system and the capabilities and opportunities afforded by software and computing technologies. The construction of requirements includes an analysis of the feasibility of the desired system, elicitation and analysis of stakeholders' needs, the creation of a precise description of what the system should and should not do along with any constraints on its operation and implementation, and the validation of this description or specification by the stakeholders. These requirements must then be managed to consistently evolve with the resulting system during its lifetime"*.

However, in practice, there is difficulty in adding these additional properties to the traditional requirement document or database and then managing them. This is because the current systems and software development paradigm generally divides the work in a project into three independent streams – Management, Development, and Test (Quality) as shown in Figure 2-2. Thus requirements engineering tools contain information related to the Development and Test streams (the requirements) while the additional properties tend to be separated in several different tools, (e.g. Requirements Management, Project Management, WBS, Configuration Control, and Cost Estimation, etc.). Moreover, activities that have become increasingly identified as being critical to success (e.g. risk planning and management) in many cases are not performed in the current paradigm. In those cases where they are performed, they tend to be treated as add-ons, and implemented in a complicated process that could have been designed and pictured by W. Heath Robinson (UK) or Rube Goldberg (USA)[69].

[69] Cartoonists in the USA and UK who drew cartoons of complicated systems designed to perform simple functions.

17.2 The object-oriented paradigm

The object-oriented paradigm however, not only provides for these add-on activities it also provides a place to store them, namely as properties within the requirement object. Consider "property" as the totality of the attribute and its value. Chapter 16 expressed "requirements" in terms of "properties and services needed" and capability in terms of "properties and services provided" where each property consists of an attribute and a value. The words functional and non-functional requirements no longer have to be used. When the system is broken down into subsystems each property (attribute and value) is allocated to appropriate subsystem elements. Traceability of properties (functional and non-functional) is built into the approach. Moreover, as described below, the packaging of processes or functionality together with the data inside the requirement object provides for automating some of the manual processes that contribute to the successful realisation of systems but which may be overlooked in the current paradigm.

17.3 Object-oriented systems engineering

The SDLC tends to begin with an undesirable situation, problem or a need. The development process begins by clarifying the need and then articulating remedying the need as a high level solution, in the form of a concept of operations of what the solution to the problem, or type of system that satisfies the need, will do. Once the functionality of the system is understood, a set of requirements for a product (the system) that will meet the need is developed. This is reviewed at the SRR, and when accepted, is implemented to realise the needed solution to the problem. Thus, the focus of the requirement in the functional paradigm is on the properties and functionality of the product to be produced.

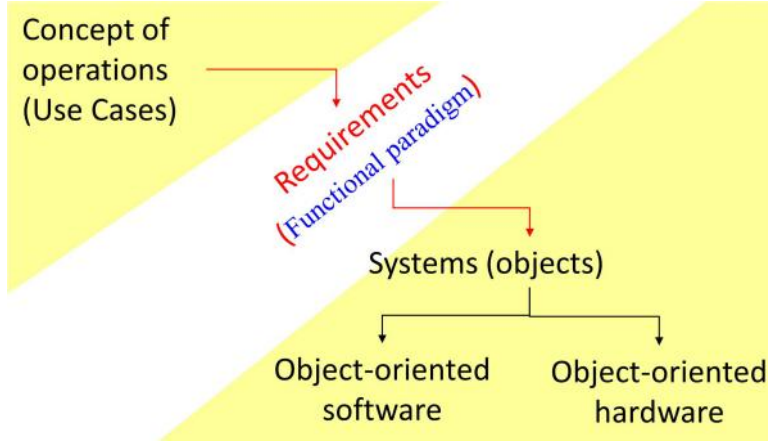


Figure 17-1 The gap in object-oriented systems engineering

Object-oriented systems engineering has a disconnection in its process as shown in Figure 17-1. Concepts of operations are stated in the form of Use Cases involving the interaction of objects, and the system is developed to implement the Use Cases via an object-oriented approach.

Requirements however remain firmly in the functional paradigm as object-oriented systems and software engineering in general, has not, applied the object concept to requirements. The general approach seems to be to either:

- Treat requirements as something that has to be produced in the early stages of the SDLC, and sometimes, adding properties of traceability and priority. Schach for example, still partitions requirements into functional and non-functional, but does add the properties of traceability and priority (Schach, 2002) page 294).
- Ignore requirements other than those that can be expressed in Use Cases. This approach in general just seems to articulate the user's needs in two redundant ways (Use Cases and requirements).

Several approaches to object-oriented systems engineering have been proposed (e.g. (Hopkins and Rhoads, 1998; Meilich and Rickels, 1999; Lykins, et al., 2000)) and there is an INCOSE special interest group working on upgrading the UML to add "systems" aspects. So from a process perspective the front end of the systems engineering process (concept of operations stating the customer's need) is object-oriented; the back end of the process (the implementation of the system to meet the customer's needs) creates physical objects so it is also object-oriented, yet the requirements still remain in the functional paradigm. Kasser and Schermerhorn wrote that systems engineers needed to use a methodology that seamlessly interfaced to the software development methodology to avoid delays and errors in translation from the system requirements to the design phase of the SLC (Kasser and Schermerhorn, 1994a). Consequently, there is a need for object-oriented requirements in an object-oriented implementation paradigm.

17.4 Research Question

The major question is "what are the properties of an object-oriented requirement? The answer is not simple. Before attempting to identify the properties of object-oriented requirements, a set of rules were established based on the maxim that a good requirement has the following three characteristics:

1. **It describes something about the physical system that will meet the needs of the customer.** This is the traditional text-based sentence that covers the functional and non-functional aspects of the system being produced.

2. **It facilitates (or rather not does not impede) the production process**. This characteristic is derived from TQM which is defined by NASA as the application of systems engineering to the work environment) (NASA, 1992b) and is concerned with the effectiveness of the production process. While requirements define a need, they can also be viewed from the contractual perspective. The cost of realising a system is based on the work and materials needed to transform needs into systems. Properties based on this characteristic include vagueness, understandability and ambiguity, namely properties that lead to cost escalations, schedule delays, or the provision of undesired functionality.
3. **It is something the customer really wants.** This is the most difficult characteristic since customers do not always to state the real requirement.

The first avenue to be explored on the journey to identify the properties is the usage of requirements in the SDLC and to explore how the object-oriented paradigm can improve the current situation.

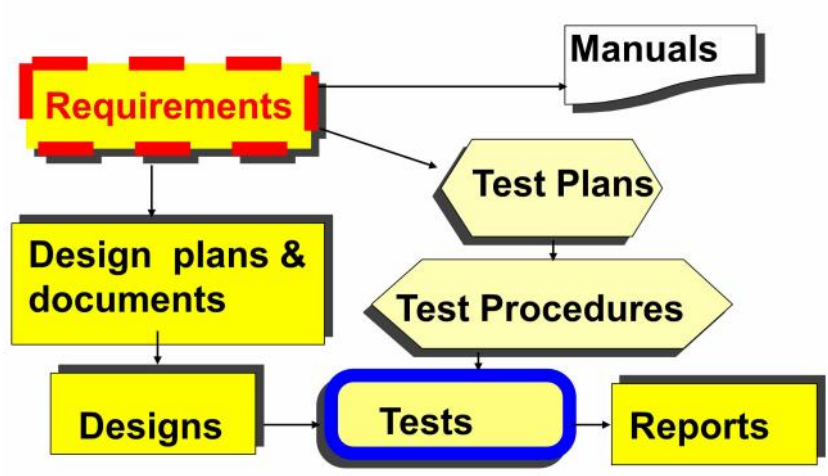


Figure 17-2 Requirements drive the work (Kasser, 1995)

17.5 Requirements drive the work

The requirements elicitation process produces a set of requirements, which represent the documentation of a function and performance of a system that meets the customer's needs. Consequently, every element of the work ought to be traceable (and chargeable) to a specific requirement or set of requirements. The work to implement a set of requirements (realise a system specified by the requirements) takes place takes place in three streams

of activities (work) shown in Figure 2-2 (management, test and evaluation, and development); hence, every requirement can be thought of as having properties driving the work in each of the three streams. This produces a view of a requirement statement as the tip of an iceberg, where the statement can be seen, but the underlying work to produce the capability that meets the requirement is hidden. An alternative view, a more traditional perspective in the form of an overview of the documentation tree, in which requirements drive the work to produce the various process-product documents, is presented in Figure 17-2.

Effectively, object-oriented requirements engineering and management not only performs the requirements engineering at the front end of the SLC, but also provides integrated information for the functions of project management, design, development, test and evaluation, and operations and maintenance as performed in the current paradigm. As in the current paradigm, the implementation work plan can be published in three documents, namely:

- The SRD.
- The SEMP.
- The systems engineering Test and Evaluation Master Plan (TEMP).

Chapter 8 documented researching the SDLC from an information perspective, and determined that these documents may be considered as summaries of the properties of the requirements in each of the appropriate stream of work. Consider the contents of each of the documents. For each document, the following properties of the requirement apply.

- **The unique identification number** – to clearly identify the requirement.
- **The text of the requirement statement** – conforming to the requirements for writing requirements (Kasser, 1995).

The other properties of the requirements are document specific as follows:

17.5.1 The system requirements document

The SRD contains the documented solution of what has to be done to provide a solution to the customer's problem (based on the OCD). The system requirement document should contain the following information for each requirement:

- **Traceability to source(s)** – where the requirement came from, i.e. the concept of operations, regulations, specific people, etc.
- **Rationale for requirement** – to communicate the reason why the requirement was included in the first place. This information is important for considering change requests during the operations and maintenance

phase of the SLC. This information is sometimes included as comments in the current paradigm, but is not required.

- **Traceability sideways** to other documents (or databases) at the same level of decomposition of the system. This provides information for use by the CCB in considering the impact of requested changes.

17.5.2 The systems engineering test plan

The systems engineering test plan drives the T&E process. The systems engineering test plan should contain the following information for each requirement:

- **Acceptance criteria** – which are provided in response to the question "How will we know that the requirement has been met by the system?"
- **Planned verification methodology(s)** - demonstration, analysis, etc.
- **Testing parameters** – the sections of the test plans and procedures that verify the system meets the requirement.
- **Resources needed for the tests** – people, equipment, time, etc.

17.5.3 The systems engineering management plan

The SEMP contains the planned resources and schedule necessary to perform the design and testing activities. The systems engineering management plan should contain the following information for each requirement:

- **Traceability to implementation** – identifies the Build in which the requirement is scheduled to be implemented.
- **The priority** of the requirement.
- **The estimated cost** to construct and test the elements of the system that provided the functionality specified by the requirement.
- The level of confidence in the cost estimate.
- **Risks** – implementation, programmatic, and any other identified. Risk mitigation approaches are an attribute of the risk.
- **Production parameters** – the WBS for the work to be done to meet the requirement.
- **Required resources** for the work, when they will be required and for how long.

Chapter 8 summarised this information in the form of a set of QSE as being necessary for effective system and software development. The QSE are not new. They are known and have been used individually in project management and systems engineering for many years. For example, the US Military Standard 2167A prescribed a set of software development folders that shall include (directly or by reference) the following information (MIL-STD-2167A, 1998):

- Design considerations and constraints.

- Design documentation and data.
- Schedule and status information.
- Test requirements and responsibilities.
- Test cases, procedures, and results.

Some of the QSE have also been incorporated as fields in requirements management tools from time to time. However, these instances seem to be the exception rather than the rule. The QSE also do not seem to have been used together in an integrated manner. The object-oriented approach integrates them. Consider the QSE as the initial set of candidate properties of requirements and thus at least improve on the current paradigm by providing a place to store those additional properties in an integrated database containing information about the process and product. Since the information in the QSE database covers the three streams of work (management, design, and test and evaluation) in the process shown in Figure 2-1, in the object-oriented paradigm, the three streams of work are considered to be interdependent not independent (Kasser, 1995).

17.6 Adding other object–oriented properties and processes to the QSE

So far the QSE database has been populated from the content of the three documents. Software engineering articulated the object-oriented approach as a way of encapsulating data and processes in ways that were not tied to physical implementations. Consider the addition of other object-oriented properties and processes to the data in the QSE database as follows.

17.6.1 Properties

The following properties could be added to the QSE:

- **Non-functional elements** of capability needed – survivability, reliability, maintainability, etc.
- **Access control property**- to control access to the requirement or selected properties. This might be used in classified situations or in corporate situations where two companies share information.
- **Version control** – identifying the version of the database.

17.6.2 Processes

The object-oriented paradigm encapsulates processes (functionality) as well as data into an object. Consider the following types of processes that might be encapsulated within the QSE database FREDIE implementation of an object-oriented requirement.

- **Text clarification**. This process scans the wording of the requirement and flag any requirements that are poorly written or do not comply with

the requirements for writing requirements (Kasser, 1995). Kasser described the First Requirements Elucidation Tool (FRED), a prototype software tool that performed this process (Kasser, 2002c). FRED produced a Figure of Merit (FOM) for a SRD. The FOM is a simple one-dimensional measurement for the quality of a document based on the presence or absence of "poor words" derived. The FOM allows comparisons to be made of the quality of documents of different sizes. The FOM is based on (Juran, 1988) and was calculated using the formula

FOM = 100 * (1 - number of defects / number of requirements).

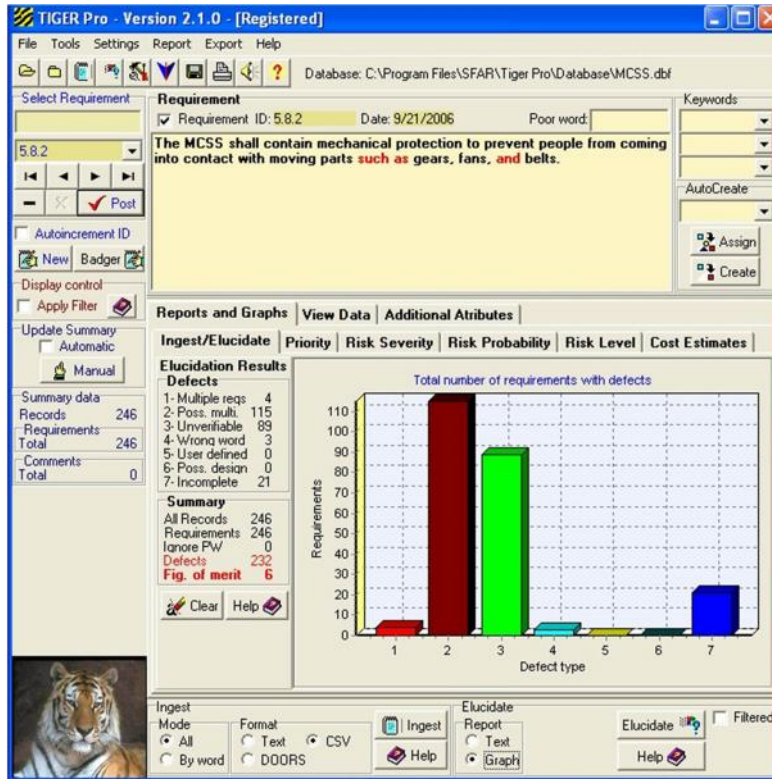


Figure 17-3 Tool to Ingest and Elucidate Requirements (TIGER)

A document without any defects achieves a FOM of 100. A document with a large number of defects can receive a negative FOM. This situation arises if requirements in the document contain more than one defect. FRED and its successor, a Tool to InGest and Elucidate Requirements (TIGER), shown in Figure 17-3, have been used in class lectures on requirements engineering in three postgraduate courses in the University of South Australia. Before TIGER was introduced, the discussions in the tutorials focussed on

the structure and format of requirements. After TIGER was introduced and used to elucidate sample requirements, the focus of the in-class discussions changed to cover the difficulties of writing good requirements. This is a significant shift in perspective (Kasser, et al., 2003).

- **Feasibility checker**. This process could check the feasibility of the requirements and flag those that were not feasible at the time they were written before they were accepted (Kasser and Cook, 2002).
- **Risk reduction and monitoring**. This process could ensure that all risks have mitigating strategies and each strategy is implemented in a WBS element, and provide appropriate warnings.
- **Property correlation.** This process could correlate properties and provide an indication when something needs further examination. For example, the process could correlate:
    - Text of requirement with acceptance criteria and identify requirements without corresponding acceptance criteria.
    - Estimated cost to implement with priority and risk. The customer could then be asked if the requirements with the following combined properties were really needed: (1) high-risk and low-priority; (2) high-cost and low-priority; (3) high-risk and high-cost? The answers may be in the affirmative, but at least the decision to implement will be an informed one.
- **Production process correctness**. This process could check for the presence or absence of other properties and provide indicators. For example, it could check that all requirements have acceptance, criteria, priorities, and cost estimates.
- **Progress monitoring.** The question "what do you mean, you can't tell me how much of my project has been completed?" is a very difficult one to answer in the current paradigm (Chapter 4). However, the use of CRIP charts can provide a better answer to the question than the measurements made in the current paradigm. They can also provide early identification of anomalies in the implementation process.
- **Facilitating and ensuring the completeness of testing**. This process could automate a manual process of building test compliance matrices. It would convert written requirement paragraphs containing multiple requirements into separate requirement paragraphs. As an example of the work that this tool could expedite, consider the following requirement (STDADS, 1992):

204.1 DADS shall automatically maintain statistics concerning the number of times and the most recent time that each data set has been accessed. These same statistics shall be maintained for each piece of media in the DADS archive.

The function would split this requirement into the following four requirements to simplify tracking the completeness of the test plans.

204.1a DADS shall automatically maintain statistics concerning the number of times ~~and the most recent time~~ that each data set has been accessed. ~~These same statistics shall be maintained for each piece of media in the DADS archive~~.

204.1b DADS shall automatically maintain statistics concerning ~~the number of times and~~ the most recent time that each data set has been accessed. ~~These same statistics shall be maintained for each piece of media in the DADS archive~~.

204.1c DADS shall automatically maintain statistics concerning the ~~number of times and the most recent time that each data set has been accessed. These same statistics shall be maintained for~~ each piece of media in the DADS archive [has been accessed].

204.1d DADS shall automatically maintain statistics concerning ~~the number of times and~~ the most recent time that ~~each data set has been accessed. These same statistics shall be maintained for~~ each piece of media in the DADS archive [has been accessed].

Leaving the sections of the requirement that were not being tested in place but stricken through clearly identifies which section of the requirement is being tested. An unfortunate side effect is that it would also clearly show the defects in the requirement. Note that the phrase '[has been accessed]' has been moved in the last two sub-requirements to clarify the sub-requirement.

17.7 Applying the concept of inheritance

Inheritance may be used in ways that extend the software engineering usage. Traceability is an inheritance function. Design elements may be traced back to requirements, regulations, etc. Expert system technology could be used to build "human experience" and tacit knowledge into the process part of the object (Kasser and Cook, 2002). This could add Artificial Intelligence to current generation requirements engineering and management tools.

Inheritance is a major advantage of the object-oriented requirements engineering paradigm since most requirements are written for systems that are similar to, or a class of, an existing system. For example, a communications satellite is a type of spacecraft, a destroyer is a type of surface ship, and a new car is similar (if not almost identical) to the previous model. Requirements reuse is becoming desirable. However, reuse is currently carried out in an ad-hoc manner (Von_Knethen, et al., 2002); the person in charge copies an old document and edits or enhances all parts they consider relevant. They integrate parts from other documents that deal with functionali-

ties they have to add. Obviously, this approach is error prone. The concept of inheritance can be applied to requirement reuse to reduce errors. For example, inheritance could be implemented as:

- A function that identifies the type of system and inherits requirements from a standard database for that type (class) of system as suggested in Chapter 16. This functionality would maximize the completeness of the requirements by ensuring that applicable non-system specific performance requirements are not overlooked.
- A function that allows selected requirements to be copied from one database to another. This functionality would save time when creating requirements for new systems having commonality of requirements with an existing system.
- Templates containing selected information for specific types of document printouts from the database.

17.8 Populating the properties of the requirement

The process of accepting requirements may be represented as shown in Figure 8-2. The customer's need (source of the requirement) is represented as a request for capability (requested requirement) and allocated an identification number. The requirement request is then assessed for priority, and cost and schedule impact on the SLC, as well as for risks and conflicts with existing requirements. The result of the impact assessment is presented to the customer who then decides to accept, reject, or modify the requirement request. However, some of these impact assessments are generally not performed in the current paradigm.

The entire set of properties cannot be populated at the same time. Population begins as shown in Figure 17-4. The initial set of properties of a requirement request submitted to the CCB consists of the:

- Requirement statement or representation;
- Source of the requirement (traceability);
- Key words;
- Rationale for the requirement;
- Acceptance criteria;
- Non-functional properties (e.g. reliability, maintainability and survivability); and
- Priority of the need for the capability.

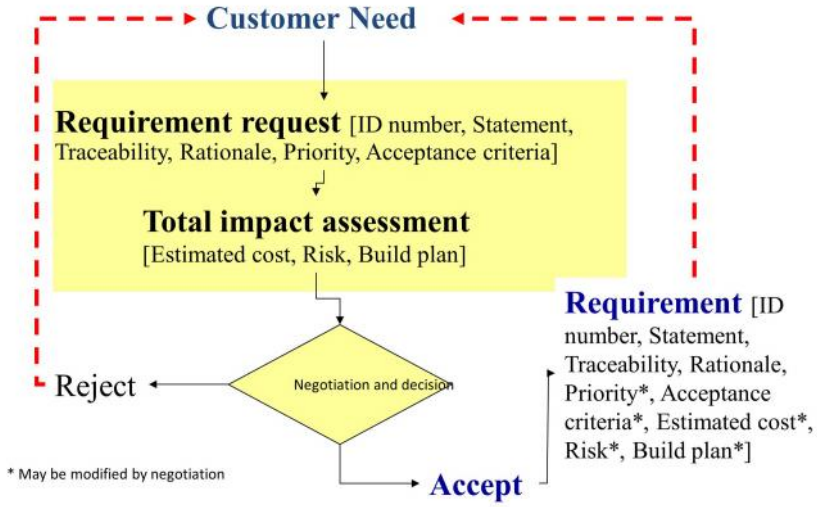


Figure 17-4 Populating the properties of the requirement

The requirement request is allocated an identification number. After performing the total impact assessment (Chapter 13), the next set of properties are populated (estimated cost to implement, risk (implementation and programmatic and their mitigation) and the Build in which the requirement will be implemented). The impact assessment is then negotiated with the customer who may accept, reject or modify the assessment. If accepted the properties are incorporated in the QSE database. If rejected, the reason for the rejection is also documented in case the same request shows up at a later time. Later on in the SLC, the test properties are populated as the test team develops the test plans etc. When the Build Plan is developed the implementation properties are further populated, and so on.

17.9 Benefits of the object-oriented approach to requirements engineering

The object-oriented approach to requirements engineering and management:

- Simplifies the SDLC by containing the add-on tasks (e.g. risk management) in the current paradigm, by definition, which if implemented at all, tends to be implemented in a complex process that could have been designed and pictured by W. Heath Robinson (UK) or Rube Goldberg (USA).
- Improves the production of well-written requirements via the use of a tool to ingest and elucidate requirements.

- Links all work back to the original requirements or to design decisions based on the requirement. The three documents discussed above are more complete since the additional properties of the requirements provide information that is generally missing in the current paradigm.
- Provides early identification of anomalies in the implementation process via CRIP charts.
- Incorporates by definition the additional properties of requirements now becoming associated with requirements engineering and management, thus avoiding the "add-on" approach of the current paradigm.
- Provides more of the information necessary for effective command and control (management) of the resources needed to realize the system than is generally available in the current paradigm. For example, the priorities of the requirements can be graphed in the form of a Bar chart as shown in Figure 17-5 and examined. If the profile looks like the one in the figure, then it doesn't seem to have much of a priority and is a candidate for cancellation.

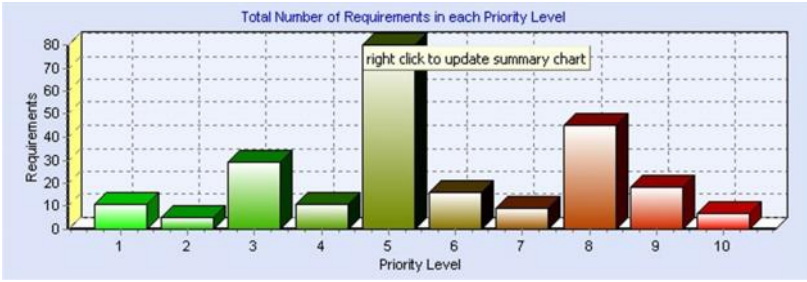


Figure 17-5 Priority profile

- Provides new perspectives on the system. For example, various properties can be examined at the SRR (before the system is built and the cost to effect changes is relatively low) and questions posed that are not easily identified or answered in the current paradigm. Typical examples are:
    - How will the customer know that a requirement has been met? The documentation of the requirement statement and acceptance criteria at the same time helps both elucidate the requirement and minimize the acceptance of requirements whose implementation cannot be verified or achieved.
    - Is a high (estimated) cost to implement, low priority requirement really needed?
    - Have the high priority requirements been assigned to early Builds? This is desirable, so that future budget cuts would tend to eliminate the lower priority performance characteristics.

- What is an appropriate Risk Profile for the type of system to be realised and is that the same profile as the instance being realised? The risks could be assigned values between 1 and 10, and the resulting profile presented as a Bar chart to provide a visible risk profile as shown in Figure 17-6. The Risk Mitigation plan would become another abstracted view of the QSE database.

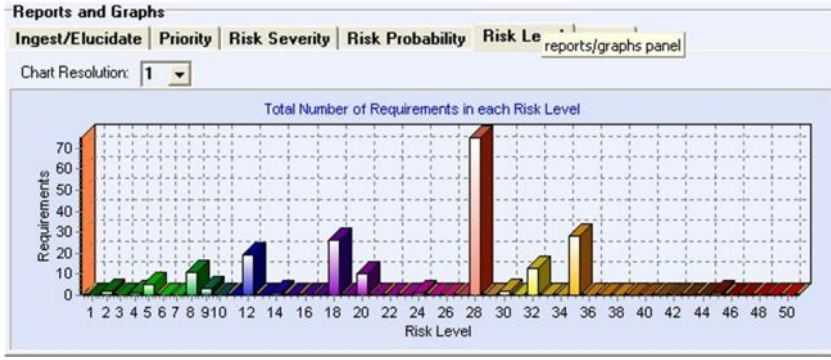


Figure 17-6 Risk profile

- Minimises loss of information and maximises correctness of information across the SLC stages by considering all documents as views of the object-oriented QSE database.
- Provides for intelligence to assist in the elicitation and elucidation of requirements.
- When coupled with the Blackboard approach used in Artificial Intelligence, allows concept demonstration tools to be developed and deployed rapidly (Kasser and Cook, 2003; Kasser, et al., 2003).

17.10 Future research

The QSE database is not the complete set of properties of a requirements object. It is only the first few steps along a journey that will determine a set, or sets, of properties for object-oriented requirements. Future research should examine areas, such as which properties are appropriate, identifying classes and methods appropriate to the process properties as well as the product properties and the nature of the Requirements Class Hierarchy.

The other avenue of research will, in the manner of systems engineering, begin with a concept of operation for how the objects could be used. Typical Use Cases might start with the examples of process functionality mentioned above, and continue to explore:

- How the properties of object-oriented requirements might be used as smart checklists.

- How requirements objects can be typed, interrelated, and re-factored, as understanding of the design implications evolve.
- How requirements objects can be elaborated by Design and Test as the project progresses, thereby generating increased confidence of operational acceptance of the system.
- If and how tools incorporating requirement objects may be used as the high-tech equivalent of the paper-based Military Standards, so beloved of the previous generation of systems engineers.
- How requirements objects might facilitate traceability from the highest level statement of need to the lowest level of implementation.
- If, and how requirements objects facilitate the engineering of complex systems.
- How object-oriented requirements can lead to object-oriented Capability Development and provide a way to manage the acquisition of Systems of Systems.
- Can tools such as CRIP charts really provide early warnings of project problems?

17.11 Summary

This Chapter has provided an overview of an object-oriented approach to requirements engineering and management together with some of the anticipated benefits. The properties of the requirement object so far identified contain data and functionality that pertain to the process as well as the product. This integration of information from the three streams of work is different to the current paradigm, which considers the three streams as independent.

17.12 Conclusions

Object-oriented requirements engineering by virtue of having many of the desirable additions to the current paradigm already built in, forces them to be addressed during the SLC and hence should be able to reduce the contribution of poor requirements engineering and management to project failures. However, object-oriented requirements engineering does not stop there. It seems to have several other advantages over the current paradigm. Further research is called for.

17.13 Recommendations

Perform object-oriented systems engineering without the use of “requirements”. The word “requirement” is closely coupled to the functional paradigm focussing on the product needed to provide the solution to the cus-

tomer's problem.

In the object-oriented paradigm, the terminology should change to correspond with the change in focus. Customer's needs should be stated in terms of capability needed. Contractors should develop and provide capability to meet the need. T&E should test the capability provided to ensure the need is met, and evaluate the capability provided to determine the performance envelope.

"Capability" can be coupled to both the product and the process that produced the product. While requirements and specifications may still be suitable for simple systems, complex systems tend to provide capability. So perhaps in the future a better term for the process of the engineering of complex systems might be Object-Oriented Capability Development.

2005

18 Reducing and managing complexity

The world has been turning to systems engineering to help manage the problems of complexity (Shinner, 1976) for at least 28 years and no practical solution is in sight. This Chapter views the system from the perspective illustrated in Figure 1-3 namely that systems are defined by their boundaries and suggests that the solution to the problems of complexity will only be found by using "out-of-the-box" thinking to change the paradigm (Kuhn, 1970) and initiates that thinking process by:

- Pointing out that the definitions of the word "system" are numerous and different.
- Hypothesising that these many definitions are formulations of problem statements and proposing a semantically loaded definition of the term 'system' from an object-oriented perspective that incorporates the hypotheses of this Chapter.
- Discussing the reasons for and implications of, the new definition and proposing that Complexity can be dealt with, and managed by redrawing the internal and external system boundaries in the context of a Simplicity paradigm.
- Sharing some perspectives on the consequences of the new definition and the insights it provides.
- Discussing a case study of a project success attributed to the Simplicity paradigm.
- Concluding with yet another definition of systems engineering, a semantically loaded one which seems to cover area of activity bounded by the HKMF (Chapter 12).

18.1 The various definitions of the word "system"

The word "system" means different things to different people. For example, Webster's dictionary contains 51 different entries for the word "system" (Webster, 2004). Consider the following representative sample of defini-

tions of the term taken from various sources from the last forty years:

- An array of components designed to accomplish a particular objective according to a plan (Johnson, et al., 1963).
- A set of concepts and/or elements used to satisfy a need or requirement (Miles, 1973).
- An assemblage or combination of components or parts forming a complex or unitary whole (Blanchard and Fabrycky, 1981).
- A number of elements and the relationships between the elements (Flood and Jackson, 1991).
- A set of different elements so connected or related as to perform a unique function not performed by the elements alone (Rechtin, 1991).
- Consists of three related sets, a set of elements, a set of interactions between the elements, and a set of boundary conditions (Aslaksen and Belcher, 1992).
- Any process or product that accepts and delivers outputs (Chapman, et al., 1992).
- The model of a whole entity; when applied to human activity, the model is characterised fundamentally in terms of a hierarchical structure, emergent properties, communication and control. An observer may choose to relate this model to real-world activity. When applied to natural or man-made entities, the crucial characteristic is the emergent properties of the whole. (Checkland, 1991).
- A network of interdependent components that work together to try to accomplish the aim of the system (Deming, 1993).
- A group of elements dynamically related in time according to some coherent pattern (Beer, 1994) page 7). Both the nature and purpose of the System are recognized by an observer within his perception of what the system does. Using this approach, models may be constructed to represent the System being studied.
- The object of study, what we want to discuss, define, analyse, think about, write about and so forth (Kline, 1995).
- Any organized assembly of resources and procedures united and regulated by interaction or interdependence to accomplish a set of specific functions. (FS-1037C, 1996)
- A collection of personnel, equipment, and methods organized to accomplish a set of specific functions (FS-1037C, 1996).
- A set of related components that work together in a particular environment to perform whatever functions are required to achieve the system's objective (Dewitz, 1996) page 5).
- A set of integrated end products and their enabling products (Martin, 1997) page 17).
- A collection of interrelated components that work together to achieve some objective (Sommerville, 1998) page 24).

- An interdependent group of people, objects, and procedures constituted to achieve defined objectives or some operational role by performing specified functions. A complete system includes all of the associated equipment, facilities, material, computer programs, firmware, technical documentation, services, and personnel required for operations and support to the degree necessary for self-sufficient use in its intended environment (IEEE 1220, 1998).
- An entity designed to function so as to achieve an objective (Westerman, 2001) page 5).
- An integrated set of elements that accomplish a defined objective (INCOSE, 2002). INCOSE then adds, "*People from different engineering disciplines have different perspectives of what a "system" is*" (INCOSE, 2002).
- A combination of interacting elements organized to achieve one or more stated purposes (Arnold, 2002).
- A bounded object which is capable of responding to external stimuli, and in response to external stimuli a system's internal components interact with each other to produce internal and external effects (Scuderi, 2004).

These definitions contain or imply the following minimum set of common elements:

- An external environment or containing system;
- An external boundary;
- Internal components;
- Relationships between the components;
- Inputs;
- Outputs.

The minimum set of common elements can also be represented by the generic diagram shown in Figure 18-1 which represents the creation of the system by taking an area of interest and drawing a boundary around that area such that anything inside the boundary becomes part of the system and partitioning the area inside the boundary into subsystems or components (Jackson and Keys, 1984). Note that Figure 18-1 does not refer to other attributes of systems mentioned in the definitions including emergent properties, purpose, and objectives.

If Figure 18-1 represents the system as defined in various ways, the question arises, why are there a number of different definitions? The different definitions seem to be a result of the way people view problems. Thus the first hypothesis in this Chapter postulates that each definition is a formulation of a problem.

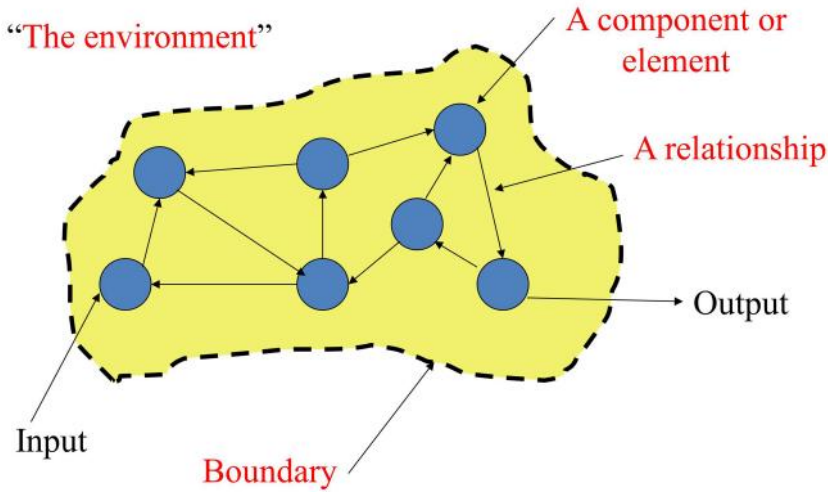


Figure 18-1 Generic representation of a system

Figure 18-1 however, is only a simple representation of the area of interest. Each component may in itself consist of components; hence the components tend to be known as subsystems. Furthermore, the representation in Figure 18-1 assumes that external elements can be ignored for the purpose for which the system was constructed. This assumption is not necessarily true as we continue to discover sometimes to our detriment. For example, in some land areas, pumping subsurface water to the surface lowers the level of the underground water table, which is not replenished by surface water seeping down after rainfall as was initially postulated, but instead results in salt water from a nearby ocean seeping sideways into the water table to maintain the level. What's more, these external effects may show up with various time delays ranging from fractions of seconds to longer than centuries (Chapter 13). Thus we change the boundaries of the system to incorporate external elements, initially considered as not having an effect on the system of interest, as and when we discover that they do in fact have an effect. The more we add to the system the more complex the system becomes.

A more generic representation of a system which includes the effects of components in or adjacent to the area of interest that affect the system is shown in Figure 18-2. Figure 18-2 reminds us of Kline's dictum that the system is only a representation of the real world (Kline, 1995), or in today's object-oriented parlance, an abstraction or a view of the real world. Kline uses the term 'Sysrep' to reflect this situation and reserves the term system to describe the area of interest from which the Sysrep is created.

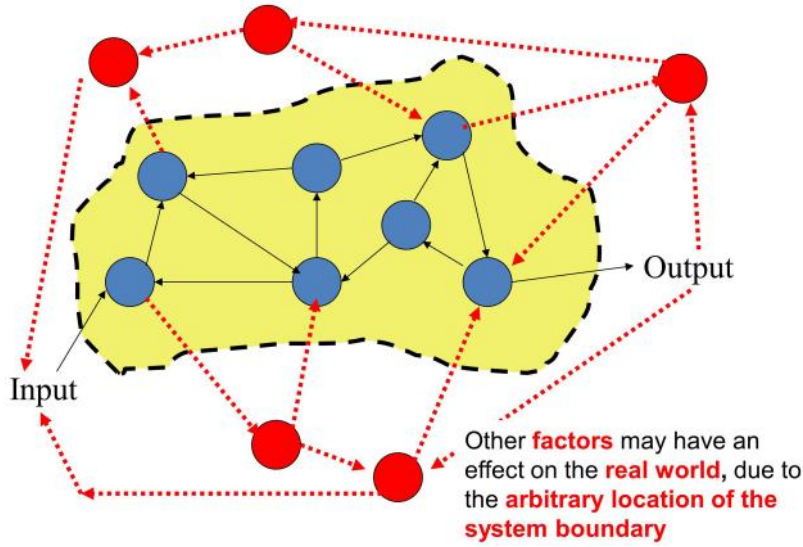


Figure 18-2 A more realistic representation of a system which takes external effects into account

18.2 Complexity is a function of poor internal boundaries

Jackson and Keys wrote "the classification of a system as complex or simple will depend upon the observer of the system and upon the purpose he has for constructing the system" (Jackson and Keys, 1984). There seem to be two types of complexity as follows:

- **Real world complexity** - in which elements of the real world are related in some fashion, and made up of components. This complexity is not reduced by appropriate abstraction it is only hidden.
- **Artificial complexity** – arising from either poor aggregation or elements of the real world that, in most instances, should have been abstracted out when drawing the internal and external system boundaries, since they are not relevant to the purpose for which the system was created. It is this artificial complexity that gives rise to complication in the manner of Rube Goldberg or W. Heath Robinson. For example, in today's paradigm, complex drawings are generated that contain lots of information[70] and the observer is supposed to abstract information as necessary from the drawings. The natural complexity of the area of interest is included in the drawings. Hence the system is thought to be complex.

[70] DODAF Operational View (OV) diagrams can be wonderful examples of artificial complexity.

Maier and Rechtin recommend that the way to deal with high levels of complexity is to abstract the system at as high a level as possible and then progressively reduce the level of abstraction (Maier and Rechtin, 2000) page 6). However, as they point out:

- Poor aggregation and partitioning during development can increase complexity (i.e. artificial complexity).
- The concept that a complex system can be decomposed into a single set of smaller and simpler units omits an inherent characteristic of complexity, the interrelationships among the components (i.e. real-world complexity).

The choice of partitioning is a major factor in the efficacy of any system description (Aslaksen and Belcher, 1992). Thus optimal subsystem boundaries must be designed for simplicity, namely:

- To abstract (hide) non-relevant information.
- To bound subsystems into self-regulating or self-sufficient entities with maximal internal cohesion of subsystems and minimal coupling between them as shown in Figure 14-9.
- That the maximum number of subsystems at any level of decomposition should generally be no more than seven (±2) to comply with Miller's Rule to facilitate understanding the system (Miller, 1956).

18.3 Cognitive filters

The process of creating a system should begin with the determination of which parts of the real-world area of interest are pertinent to the situation. The next step is to abstract out the non-pertinent elements to create the system or Sysrep. This is the problem formulation process. The abstraction process uses a filter to separate the pertinent from the non-pertinent. This filter is known as a "cognitive filter". Cognitive filters are filters through which we view the world. They include political, organizational, cultural, and metaphorical, and they highlight relevant parts of the system and hide (abstract out) the non-relevant parts.

Thus the various definitions of the word 'system' reflect views of problems through different cognitive filters by the creators of the definitions, namely systems engineering in the manner of Wymore who writes that systems engineers are problem starters (Wymore, 1993) page 2). Moreover, once systems are defined they take on a life of their own. Forgotten is the reason for which they were created, and subsequent problems are viewed through the same cognitive filter, and, since these new problem situations may not exactly fit the cognitive filter, the problems are adjusted to fit the filter, namely the solution defines the need.

In some instances, cognitive filters can also add material that hinders solving the problem[71]. For example, the paradigm that requires the presence in a system of the following two characteristics (emergence and purpose) that were not shown in Figure 18-1 is a cognitive filter.

18.3.1 Emergence

Checkland and Scholes in a discussion on systems thinking state that "a complex whole may[72] have properties which refer to the whole and are meaningless in terms of the parts which make up the whole. These are the so-called emergent properties" (Checkland and Scholes, 1990) pages 18-19). Emergent properties can be intentional and unintentional (Watts and Mar, 1997). There seem to be three types of emergence:

- **Undesired** – functionality performed by the system that is undesired, also known as 'side effects'.
- **Serendipitous** – beneficial and desired once discovered, but not part of the original specifications.
- **Desired** – being the purpose of the system and can only be achieved by the combination of the subsystems or components. This is a reworded version of Rechtin's definition of a system (Rechtin, 1991) quoted in Section 18.1. Examples of desired emergent properties are the functionality performed by the system, system reliability, system weight, system safety, and system usability. Note the use of the word "system" in each of the examples because they refer to a value that can only be achieved by the system as a whole.

Figure 18-1 and Figure 18-2 do not contain an explicit element that represents 'emergence' or 'emergent properties'. Checkland and Scholes in introducing the concept of emergent properties in a discussion on systems thinking twice state that a *complex whole **may**[73] have emergent properties* (Checkland and Scholes, 1990) pages 18, 19 and 25). Emergence is a characteristic of the system, thus the notion (or requirement) that **systems must have emergent properties** is a cognitive filter[74]. Systems without emergent

[71] For example, the differences between the Catholics and Protestants in Northern Ireland are major to many of the inhabitants of the country, but are hardly noticeable to most of the rest of the world.

[72] Note the use of the word "may", not "must"!

[73] This author's emphasis of the word "may".

[74] Consider this example of the cognitive filter in action. While researching this Chapter I pointed out to a colleague that the original Checkland and Scholes texts on emergent properties quoted above used the term "may" not "must". There was a pause, a blank look, and then the shutters seemed to come down behind the eyes.

properties can be represented graphically in the following manner. The system boundary defines the functionality of the system. This functionality is mapped onto components (subsystems) as part of the design process in creating the physical system as shown by the coloured areas in Figure 18-3. However, some of the functionality may be a result of the emergent properties of the system. This can be represented in a geometric format as:

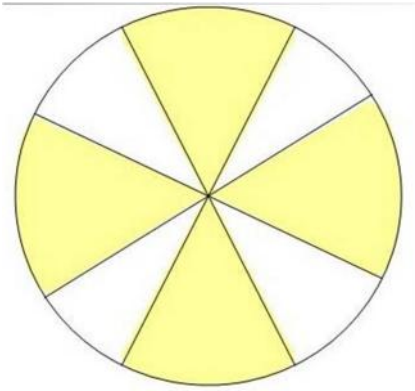


Figure 18-3 Properties of a system

- Functionality = Area of system.
- Area of system = (sum of functionality of all components + area of system not in a component).
- Area of system not in a component = emergent properties.

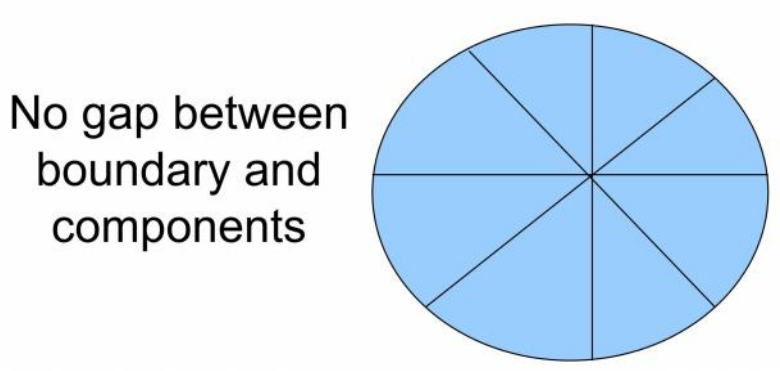


Figure 18-4 Representation of a system without emergent properties

---

My colleague restated "*systems must have emergent properties to be considered systems*!" End of conversation.

The emergent properties lie within the area of the system outside the components or subsystems. By moving the boundaries of the subsystems (incorporating more capability or functionality into the subsystems) the system could also be drawn such that there are no gaps between components as shown in Figure 18-4 and hence produce a representation of a system without any emergent properties.

18.3.2 Purpose

Some systems don't have a purpose that we know of. For example, what is the purpose of the solar system? Thus the notion that systems must have a purpose likewise is a cognitive filter. Purpose can be considered as desired emergence as stated above. For example, a system containing four walls and a roof is a dwelling. Each component cannot perform the function of a dwelling on its own. The capability to function as a dwelling is the emergent property of the system made up by the walls and roof. It is also the purpose of the system. The purpose is not in the system, it is in the mind of the person drawing the boundary that creates the system because people draw boundaries (Churchman, 1979) page 91).

18.4 Introducing Simplification

This Chapter now hypothesizes that unnecessary excessive complexity is incurred by fitting a solution to a problem by the process of adjusting the area of interest in the real world to fit the cognitive filter[75]. Recognising that excessive complexity is a symptom of an underlying problem within the foundation of the current paradigm (Chapter 3), this Chapter proposes that what needs to be done is instead of fitting a solution to the problem, and creating a single (complex) system, we need to formulate the correct problem correctly and create an appropriate set of simple systems (from different viewpoints) as described below.

Consider the perspectives shown in the modified representation of a system in Figure 18-5. Aslaksen writes, *"Our choice of boundaries and interactions depends on what we are trying to understand and what we, as engineers, want to achieve by this understanding, so that system definitions are inherently subjective. In effect, defining a system is the first step in creating a model of some part or aspect of reality"* (Aslaksen, 2004).

The internal view shown in Figure 18-5 represents the "you can't see the forest for the trees" aphorism as well as demonstrating the reason for Kline's assertion that "*neither the top-down synoptic view nor the bottom-up*

[75] This process is also known as "pattern matching" in the parlance of the object-oriented software paradigm.

reductionist view can, by itself, supply reasonable understanding of systems with hierarchical structure incorporating interfaces of mutual constraint and multiple levels of control" (Kline, 1995).

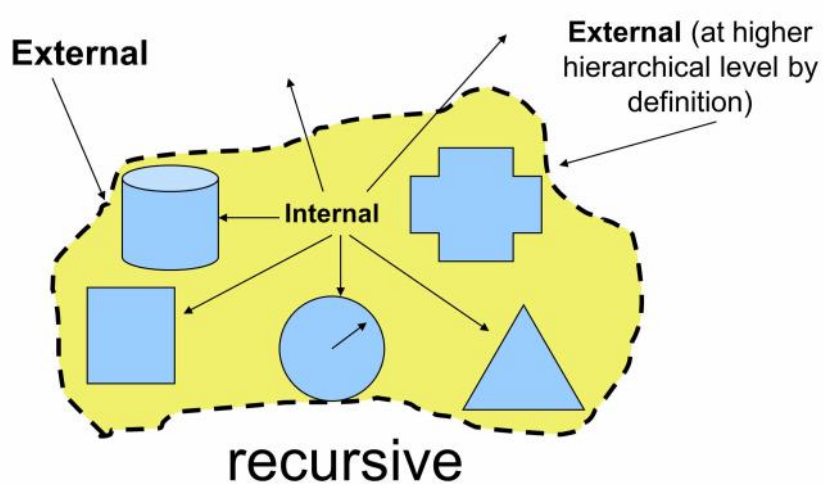


Figure 18-5 Internal and external views of a system

Furthermore, it shows that a complete view of any part of the system cannot be obtained from any single viewpoint inside or outside the system. Thus the use of internal views without any external views does not provide a complete understanding of a system and can lead to unnecessary complicated solutions to the wrong problem.

For example, consider the hierarchical view of a Meta-system shown in Figure 18-6. When one of its subsystems, System B, is viewed externally from above, it is seen to be made up of three sub-systems. Yet when the three sub-systems of the Meta-System (Systems A, B and C) are viewed internally and externally from a horizontal viewpoint, the horizontal view through the subsystems is commonly referred to as a System of Systems. Since the process of constructing systems from sub-systems ranges through many levels of hierarchy (from sub-atomic particles, though atoms, molecules, etc. all the way up to the universe), the complexity assumed for Systems of Systems seems to be a direct result of the lack of external views of the system comprising the Systems enclosed in a boundary now defined as a System of Systems (and the high degree of coupling (interaction) between the subsystems). These external views are also at a higher hierarchical level than the component of the system, and must also be via various appropriate cognitive filters. Moreover, each filter must provide a view (information) that does not overlap the information provided by the other views. Thus in order to obtain a complete understanding of a system, there must be a number of internal and external viewpoints such that:

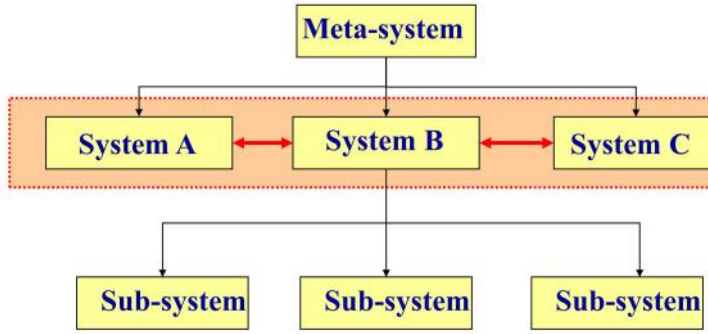


Figure 18-6 Hierarchy of systems

- Each is separated in the space domain, so that no relevant part of the system is hidden and,
- Each is simultaneous or synchronized in the time domain.
- Moreover, if the behaviour of a system over time is being examined, a series of simultaneous space-separated, time synchronous views are required in order to obtain an understanding of the system.

18.5 Yet another definition of the term "system"

This Chapter now proposes a semantically loaded definition of the term 'system' that incorporates the hypotheses of this Chapter, namely:

A system is an abstraction from the real world of a set of objects, each at some level of decomposition, at some period of time, in an arbitrary boundary, crafted for a purpose.

Consider the implications of the terminology used in the definition.

- **Abstraction** - used in its object-oriented meaning to remind us that a system is not the real world, but is Kline's Sysrep and must always be viewed in that context.
- **A set of objects** - the components of the system.
- **Each at some level of decomposition** - each component may itself be an aggregate of components.
- **At some period of time** - not only do the system and its components have to be considered at the same period of time (Beer, 1994), but consideration has to be taken into account that the area of interest represented by the system may change over time.
- **In an arbitrary boundary** - the boundary is crafted by the observer to enclose a section of the real world (Jackson and Keys, 1984). The word

arbitrary is used because the boundary may appear arbitrary to other entities until the purpose for drawing the boundary is understood. The act of drawing an external boundary implies a containing system, so the concept of hierarchies is built into the definition. The act of drawing internal boundaries or partitioning the system defines the components or subsystems. The choice of partitioning is a major factor in the efficacy of any system description (Aslaksen and Belcher, 1992).

- **Crafted for a purpose** - this is the part of the definition that really changes things. The system does not have to have a purpose. The boundary is defined by the purpose for which it is drawn in the mind of the observer as implied in the definition of a system by (Beer, 1994) quoted in Section 18.1.

18.6 Perspectives

This definition introduces new possibilities, currently being researched, some of which are discussed below.

18.6.1 Bridging the gap between hard and soft systems

The definition of a system proposed above seems to include or cover the various definitions of the term 'system' listed above obtained from the hard and soft systems literature. If this statement is correct, and the definition can cover most, if not all, of the other existing definitions of systems, then this new definition applies to both hard and soft systems and could serve as a baseline for bridging the gap between hard and soft systems.

18.6.2 Open systems

Inputs and outputs cross system boundaries when the purpose of drawing the system boundaries has to do with the inputs, outputs, and the internal elements inside the system boundary. This is the traditional textbook open system.

18.6.3 Closed systems

The typical textbook examples are (pendulum) clocks and the solar system. However, the real world does not contain any closed systems. In the real world, clocks operate in a gravity field and the solar system is bathed in external radiation. However, when a boundary is drawn for the purpose of developing an understanding of the way the internal components of a clock or the solar system work together, then these systems can be considered as closed systems for that specific purpose.

18.6.4 Relationships between components

A set of objects does not become a system until a boundary is drawn around the set by some entity for some purpose. For example, two separate audio amplifiers on a circuit board do not constitute a system until a boundary is drawn around them to include them in a stereo system. To repeat, it is the act of drawing the system boundary that creates the system.

18.7 Simplifying the process of systems analysis

The process of systems analysis is supposed to be a way of simplifying the complexity of the real world to allow us to manage the area of interest. This process however, instead of simplifying matters has tended to introduce complexity. As systems become more complicated (artificially complex) we add extra layers of control which makes the system even more complicated. This is a positive feedback situation.

The process of systems analysis starts by creating a system to represent the area of interest. We do this by abstracting a single system (Sysrep) with a fixed boundary from a real-world area of interest. Once the system is defined, further views are generated of the system by abstracting out non-relevant information via various cognitive filters as shown in Figure 18-7.

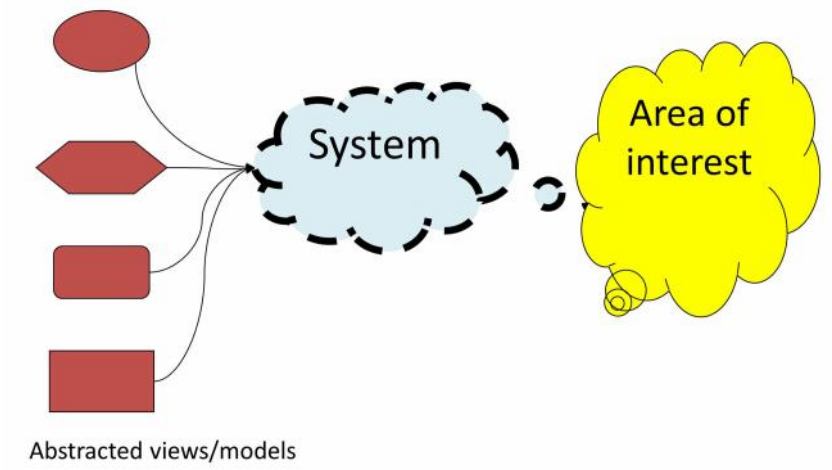


Figure 18-7 Abstraction process leading to complexity

The complexity needs to be abstracted out when the drawings (Sysreps) are produced to achieve Simplicity. For example, consider the US Space Transportation System (Space Shuttle) and the International Space Station (ISS). Each is a complex system in itself, yet when solving the problem of docking a Shuttle to the ISS, all the underlying complexity that is not relevant

to the docking problem is abstracted out. Thus, we construct a closed system to simplify the problem by abstracting out (filtering out) everything other than information pertinent to the:

- Relative positions of the spacecraft.
- Relative velocity of the spacecraft.
- Relative alignment in X, Y and Z orientation.

This is an instance of Simplicity. Why can't we use the same paradigm elsewhere? We need to learn how to view things differently, namely adjust our cognitive filters for Simplicity not Complexity.

In object-oriented parlance, the real world is a data source; we deal with abstracted views of that data source. One specific view does not fit all purposes. The use of one specific system (view) for all purposes introduces unnecessary complexity, since we have to fit all activities to that one view.

Abstracted views are systems. Why not just abstract multiple views to help with the purpose directly from the real-world area of interest? Each abstracted view can be kept simple to facilitate its purpose, and a number of abstracted views will be needed for a complete understanding of the area of interest. For example, the different views of the SDLC in Figure 9-1 and Figure 5-2 abstract out the pertinent information that is relevant to the aspect of the SDLC being discussed in association with the Figure. In the world of Simplicity, there is no such thing as a (single) system that represents an area of interest. There are instead a number of systems, each of them dealing with some aspect of the area of interest in the manner shown Figure 18-8. Consider the examples of a rock, a camera and a human being.

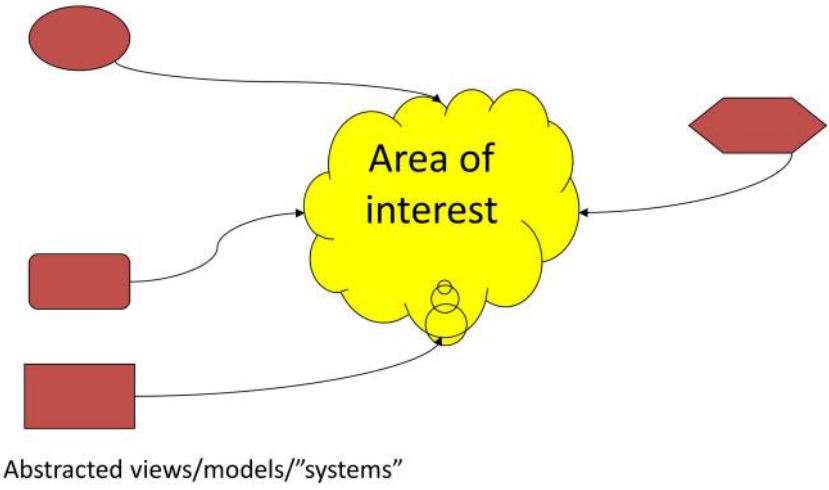


Figure 18-8 Abstracting various views directly from the area of interest

18.7.1 A rock

As an example, consider the investigation of a rock. The system boundary is drawn at the surface. While determining the nature of the rock, various views can be used including:

- **Sight** – one looks at its colours.
- **Taste** – taste might give us some information about the chemicals in the rock.
- **Weight/mass** – might tell us something about its composition.
- **Touch** – the surface texture might be of interest.
- **Chemical analysis** – the components might be of interest.
- **Radiation** – could tell us something

Each view provides information that the others do not, helping to build up a complete understanding of the nature of the rock.

18.7.2 A camera

Consider a camera. When we consider the device that takes the photograph, we draw the system boundary around the camera. However, when we consider the act of taking the photograph the boundary is redrawn to include the photographer. When considering transporting the camera the boundary is drawn to include the transportation elements. Developing one representation that includes all the elements for photographing and transportation and then requiring the elements under consideration for a specific situation to be abstracted out of the representation, creates unnecessary complexity. The three separate simpler views, abstracted out of the real world are simpler for understanding the various aspects of the use of a camera in photography.

18.7.3 A human being

Some areas of the real world can only be fully understood by examining the internal components of the system and observing it in action in its environment. Consider a human being, a biological system. To learn about the interaction between internal subsystems we may have to observe the sample in action in specific situations and either observe or infer the interaction. To learn about the internal subsystems we have to dissect a sample of the system. Once dissected, an individual example cannot usually be restored to full functionality. However we have learnt something about the class of systems it represents which can be applied to other instances (human beings).

18.8 Complexity vs. simplicity

Is the world more complex today than in previous times, or is there just a perception that it is so due to unnecessary complexity? Within a given area

of interest, different people may draw different system boundaries (INCOSE, 2002). In the current paradigm that tends to happen for a number of reasons which include communications failures. In the Simplicity paradigm most of the reasons might be stated as being due to the use of different cognitive filters.

"A simple system will be perceived to consist of a small number of elements, and the interaction between these elements will be few, or at least regular. A complex system will, on the other hand, be seen as being composed of a large number of elements, and these will be highly interrelated" (Jackson and Keys, 1984). The complex system seems to be an example of:

1. Poor boundary drawing between the system components (subsystems) since the boundaries should be drawn to achieve high cohesion in any given component and low coupling between the components (Ward and Mellor, 1985).
2. The use of a single abstraction (partition of a system into subsystems) for all activities associated with the area of interest instead of using multiple abstractions as appropriate since at any one time we are NOT studying all aspects of the area of interest represented by the system.

System and subsystem boundaries depend on the prerogative of the situation and are dynamic, not static. Unnecessary and the perception of unmanageable complexity is a result of the use of a single static system boundary for all purposes. Simplicity depends on the use of multiple views or representations of an area of interest and each view abstracting out all the information not pertinent to the view. This is similar to:

- The concept behind the 26 standard views prescribed for the DODAF (DoDAF, 2004).
- The use of the wave theory (Huygens, 1690) to explain some aspects of electro-magnetic radiation and the particle theory (Newton, 1675) to explain other aspects[76].
- A plumber examining the system, replacing a washer and charging $100 for the job. When asked if $100 wasn't a little bit excessive for replacing a washer, the response was "replacing the washer cost you one dollar, the other $99 was the charge for knowing which washer to replace." In electronic engineering terms this is a signal to noise filtering problem. The relevant information is the signal, the remaining information the noise. A good filter (a combination of knowledge, experience and ex-

[76] Mind you this could also form the basis of an argument that there is no underlying theory of electronic-magnetic radiation.

pertise) will abstract out the pertinent information and simplify the problem.

Note both Simplicity and Complexity can suffer from the problem in which information pertinent to the system is not abstracted out from the real-world area of interest (forgotten). Simplicity is an object-oriented approach and relies on the concept of inheritance of views from the same class of systems to alleviate this problem. The object-oriented world has taught us that boundaries are in the mind of the observer. For example, think of your favourite sports. Picture the game in progress as you look out over the playing area.

- How many objects can you see? People, bats, balls, clothes, etc.
- How many classes of objects? Sweater is an instance of the class of clothes; bat and ball are instances of the class of sport equipment, etc.
- What is the difference between an object and a class?

18.9 Case Study Luz SEGS-1

The LuZ SEGS-1 system (Chapters 11 and 22) also an example of how the use of a set of different cognitive filters (out-of-the-box thinking) was a major contribution to a successful project. The initial traditional hardware-software based design approach was for a conventional central minicomputer design. The central minicomputer would act as the human interface to the system, perform the pointing position calculations for each mirror, and then control all of the mirrors via a high-speed data link and the LOCs at each mirror. Ethernet was proposed for the data link, but was still in its infancy in those days, and was expensive. The LOCs would position the mirrors based on control information from the central processor, and collect pointing angle and oil temperature information from each mirror. This design was complicated and high risk and the future of the start-up company was riding on meeting the development and installation schedule. Recognising the futility of the conventional approach, the problem was formulated differently77. The approach can be stated in several ways:

- Out-of-the-box thinking was employed.
- A pattern match was made to a fleet of earth orbiting spacecraft and a central ground station. The LOCs were the analogue of the spacecraft and the central processor, the ground control station. The data flows

[77] One example of the benefits of formulating the problem in a different manner was discussed in Section 4.6.

between the LOCs and the central processor were telemetry, tracking and control data flows.

- An alternative design was chosen employing a different set of cognitive filters.

The traditional approach drew a boundary between hardware and software. The hardware subsystems would interface the mirrors, sensors and motors to the central processor via simple circuitry in the LOCs; the software subsystems would reside in the central processor, a minicomputer, and perform the control functions.

In this case, the subsystem boundaries were not drawn between hardware and software. An object-oriented functionality-based approach was chosen instead. The control functionality was distributed throughout the system in embedded microprocessor firmware as well as in the central processor, which meant functionality was replicated in the LOC, situated at each mirror (in the form of several software components) instead of existing only in the central processor. The system architecture took the form of self-regulating systems (LOCs) communicating with the CCS (control element) as shown in Figure 14-9 and implemented as described in Section 11.8.1. The rules for drawing the subsystem boundaries were as follows (Kasser, 1999):

- Minimize coupling and maximize the cohesion of the subsystems.
- Consider the operator as part of the system.
- Use self-regulating subsystems.
- Use railroad buffers for signal passing.
- Use a testable software architecture.

Consider each in turn.

18.9.1 Minimize coupling and maximize the cohesion of the subsystems

This approach is based on the Ward and Mellor methodology (Ward and Mellor, 1985) and fits nicely into both object oriented and conventional design approaches.

18.9.2 Consider the operator as part of the system

Consider the operator as part of the system - This approach allows early Builds of a system to perform functions manually, and then provides automated capabilities in subsequent Builds (either as part of the Build Plan or as planned upgrades as more is learned about the system's behaviour). It also allows subsystems to be coupled in a well-defined and understood manner. For example, one system may act as "the operator" for a second subsystem.

18.9.3 Use self-regulating subsystems

Partition the subsystems to perform as self-regulating subsystems carrying out its functions in a self-regulating manner. The rules for (minimizing) coupling and (maximizing) cohesion must be observed when partitioning the subsystems. The subsystem should contain the appropriate feedback circuits to allow it to perform its task until subsequent instructions are received. It transmits status information about itself, and receives command instructions from other subsystems as shown in Figure 14-9.

18.9.4 Railroad buffers for signal passing

This was a key element to the success of the project. All signals were passed between processes via buffers at both ends of the interface as shown in Figure 18-9. Software modules were not allowed to build and transmit messages on the fly, or react to messages as they are received. The term "railroad buffer" is used because the interface area of the software component (subsystem) looked like a freight yard at a railroad station. This element allowed modules to be tested in both a static (standalone) and a dynamic manner. The interface was tested by placing known data in a transmitter buffer and ensuring the data appearing in the corresponding receiver buffer was correct after the necessary event which triggered the transfer. The modules were tested by placing data in the receiver buffer, and initiating the processing task. The data in the output buffer or the state of the module was then checked to see it met the specifications for the processing task. This element has much in common with client-server techniques, but may cause a small and in most cases an unnoticed loss in performance. These buffers may also be considered as the software equivalent of hardware test points.

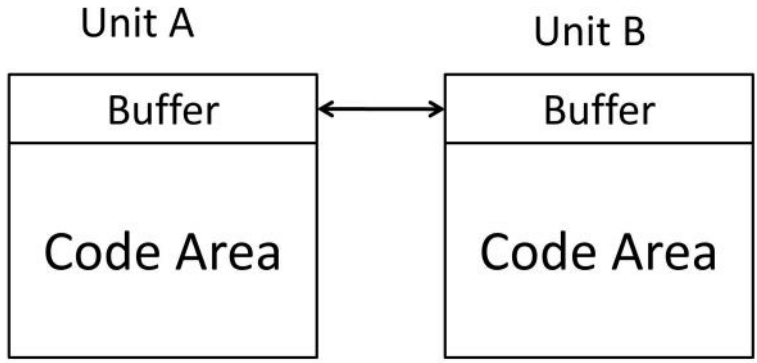


Figure 18-9 Railroad buffers for signal passing

18.9.5 Use a testable software architecture

The software architecture employed in the central processor using the principle of controlled self-regulating subsystems passing information via railroad buffers was as shown in Figure 18-10. These subsystems were the:

- **User interface** - The data display and entry device(s) which interacted with the users. The user interface was developed by a single group which ensured consistency.
- **Algorithm executor** - The top-level subsystem which carried out the work the system was built to perform.
- **Database(s)** - The database components of the system.
- **Operator Window** - A window into the system. It displayed all status, alarm, error and event states, and the contents of buffers.
- **External interfaces** - The interfaces to the external elements to the system.

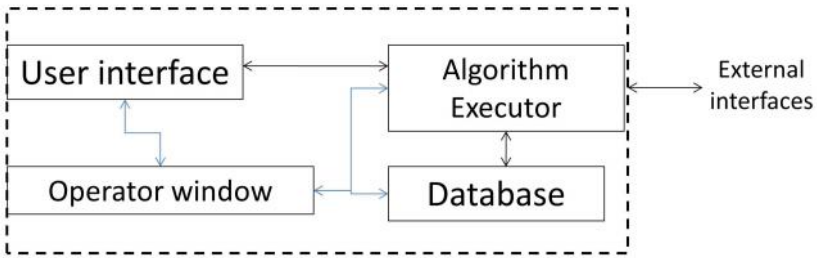


Figure 18-10 Testable software architecture

The operator interface also served as the major troubleshooting tool both during system commissioning and operational troubleshooting. By allowing the code produced to display the state of the system during the development process to be left in the final product (with appropriate documentation), this feature reduced the need to develop custom tools to test portions of the system, hence reducing the cost of the system This architecture is applicable to the generic class of systems that can be built from the user/operator interface inwards.

18.9.6 Summary so far

In summary, applying systems thinking and viewing the system through several different cognitive filters (including an embedded software approach) instead of the classic hardware-software approach meant that the pointing functionality was transferred from the central processor to the individual LOCs. This allowed the planned mini-computer to be replaced by a Z-80 eight-bit microcomputer system costing $2,000, avoiding at least $900,000 in minicomputer hardware and software costs. In addition if the control link

failed for a short period of time, the mirrors would continue to point at the sun and generate heat.

18.10 Redrawing the contractor sub-contractor boundaries in certain types of Defence contracts

This section contains an example of how redrawing boundaries could simplify and lower the cost of the Small and Small Disadvantaged Business (SDB) set asides in the US Government contracting process while retaining the socio-economic benefits to society for certain types of contracts.

18.10.1 Background

The US Federal Acquisition Streamlining Act (FASA) of 1994 established a 5% Small and SDB Set-Aside goal for civilian agencies. However, the socio-economic benefits of these goals were hard to achieve for the following reasons.

- By the mid 90's concerns had been raised regarding the quality of the products provided by Set-Aside contracts. These concerns were so serious that the whole Set-Aside program had been questioned. As a result, the DOD went as far as suspending its (rule of two) SDB Set-Aside program in October 1995.
- While the agencies required prime contractors to propose subcontracting plans, overall they did not seem to follow up on the plans and require compliance. In such circumstances, the plan was an element to be checked off for the proposal and subsequently forgotten.
- Many Small and SDBs do the same kind of low-capital-cost knowledge-intensive types of work namely Systems Engineering and Technical Assistance, Business Process Reengineering, software engineering, and operations support activities as the large contractors. Thus when large companies identify subcontracting partners there tends to be a degree of overlap between the functions each does on the contract. This overlap allows the large prime contractor to squeeze out the subcontractor as much as possible.

18.10.2 The problem

About 25 years ago, according to the US Small Business Administration (SBA), studies had shown that small companies were more cost effective and innovative than large companies[78]. Since many Small and SDBs perform the

[78] In 1982, there were 2.4 times as many small firm innovations as large firm innovations per employee according to the (SBA, 1982)

same kind of low-capital-cost knowledge-intensive activities as these large contractors, they could form strategic alliances in a multiple-award-task-ordered (MATO) contract environment to compete for these contracts. Such strategic alliances could:

- Provide the Small and SDB team partners with more work than the overall Small and SDB set-aside percentage.
- Reduce the need for Small and SDB Set-Asides.
- Reduce the complaints of "unfair" by non SDBs
- Give the Government lower cost work of equal if not better quality than large companies for these types of contracts.

With these apparent reasons to form strategic alliances, consortiums and teams, most Small and SDBs still, in general, tended to:

- Attempt to team with large companies in a prime-subcontractor relationship to meet the contractual Set-Aside requirements.
- Concentrate their marketing efforts on Small and SDB Set-Aside contracts.

The reasons why those strategic alliances were not formed was focus of doctoral research (Kasser, 1997).

18.10.3 The traditional boundaries

By the mid 90's the Government had begun to award MATO contracts in an attempt to obtain quality products at reduced cost by removing the monopoly effect present after a single contractor won a multiple year award. The traditional MATO contract award pool scenario is shown in Figure 18-11. Several contractors bid on the Government's RFP and a selected subset qualifies for the award pool. The subset that qualifies then competes for individual tasks within the contract. Each contractor has its Small and SDB subcontractors as required by the contract. This arrangement:

- Allocated the Small and SDB funding goals in a horizontal manner. The Small and SDBs are below the large prime contractors in the hierarchy of the contract.
- Is a way of providing socio-economic benefits to the Small and SDB community.
- Loads the cost of the contract with the overhead to plan and administer the regulatory required Small and SDB subcontracts as well as the subcontract pass-through costs.

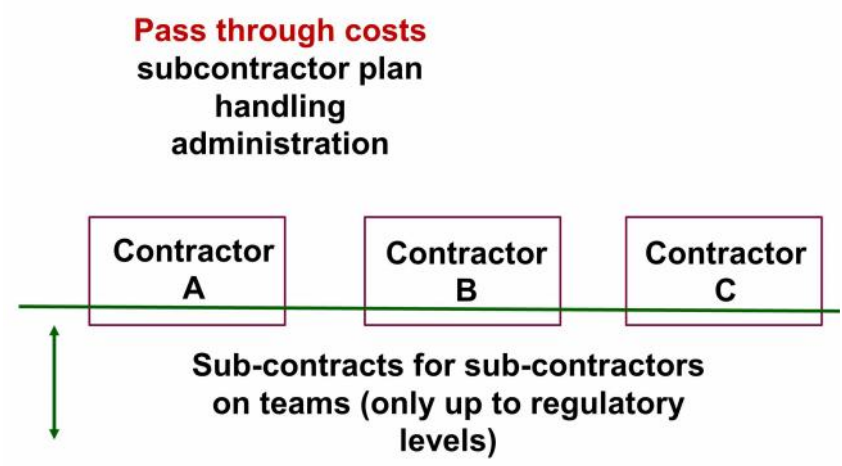


Figure 18-11 Multiple award set aside scenario

18.10.4 Redrawing the boundaries

Redraw the organizational boundaries between the contractors from the horizontal contractor-subcontractor relationship to a vertical structure in which Small and SDBs joined in a strategic alliance to form a virtual large contractor with the following contractual changes:

- Make the guaranteed minimum for the procurement equal to the Small and SDB Set-Aside dollar amount.
- State that one or more awards will be made to a Small or SDB or alliance thereof.
- Completely remove the Small and SDB subcontract requirements from the RFP.

In this situation, the Small and SDB arrangement is a vertical arrangement as shown in Figure 18-12 rather than the current horizontal arrangement.

18.10.5 A benefit of the simplicity approach

The vertical arrangement eliminates the:

- Overhead costs to plan and administer regulatory required Small and SDB subcontracts.
- Subcontract pass-through costs.

The vertical arrangement would also seem to provide the Government with a win-win situation from two perspectives, namely:

- The Small and SDB perspective.
- The large business perspective.

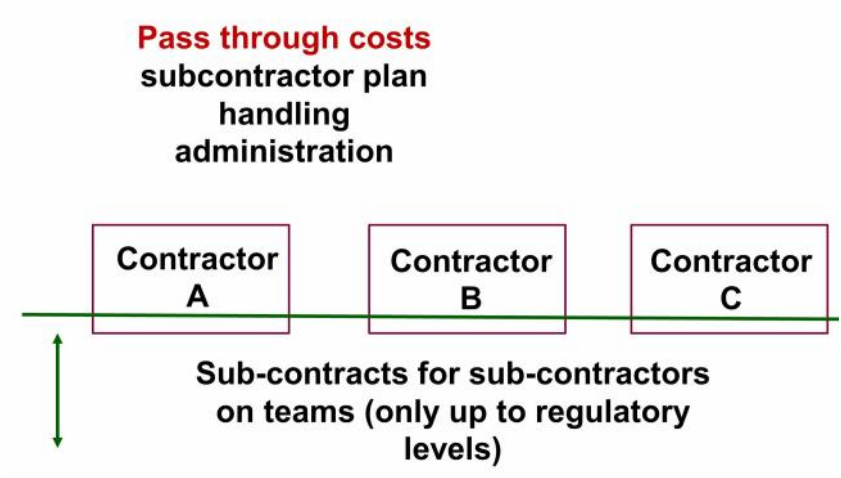


Figure 18-12 Vertical set-aside scenario

Consider each of them in turn.

18.10.5.1 The Small and SDB perspective

Their performance would affect the cost of the contract as follows. If they were to be:

- **Outstanding** - Their technical approach and low cost would win them a larger percentage of the work. The Government would get more value added for each dollar spent, and the total amount spent on Small and SDBs would increase, helping the Agency meet its Small Business dollar goal in a cost-effective manner.
- **Mediocre** - They would receive the Set-Aside minimum percentage of the contract and some help in improving their performance.

18.10.5.2 The large business perspective

The cost of work would be lower for the following reasons:

- **Proposal preparation costs** - By eliminating the need for the large companies to develop a subcontracting plan.
- **Contract performance costs** - By reducing duplicate management structures in each company, and eliminating the prime contractor's subcontract management functions.

In addition, outstanding Small and SDB performance would drive down the cost of the contract, with no reduction of quality, because either the Small and SDB capture all the work, or, the large business would have to improve to remain competitive.

18.10.6 Summary and conclusion

Redrawing MATO contract contractor boundaries into a vertical set-aside approach may be a way to provide the socio-economic benefits to Small and SDBs without giving up quality or incurring additional costs in carrying out the Small Business regulations. Implementation of this approach would however probably require a change to the Federal Acquisition Regulations (FAR).

18.11 Yet another definition of systems engineering

The word system has been defined above. The American Heritage dictionary defines the word "engineering" as "*The application of scientific and mathematical principles to practical ends such as the design, manufacture, and operation of efficient and economical structures, machines, processes, and systems*" and "*The profession of or the work performed by an engineer*" (American Heritage, 2000).

Thus by combining the two definitions, we have **Systems engineering is the application of scientific and mathematical principles to the abstraction (from the real world) of a set of objects, each at some level of decomposition, at some period of time, enclosed in an arbitrary boundary crafted for a purpose**.

This definition of systems engineering states that **systems engineering is an activity**, namely the application of scientific and mathematical principles, and seems to cover all five layers of systems engineering at all points within the two dimensional space of the HKMF defined in Chapter 12. The definition does not state anything about the nature or purpose of the "application" or the necessity to meet anyone's needs. To this author, at least, it is more realistic than a definition that includes goals and purposes. The latter type of definition is an ideal to be aimed at, and not a pragmatic portrayal of the real world.

18.12 Summary

This Chapter has discussed ways of dealing with complexity by changing cognitive filters and redrawing boundaries.

18.12.1 The hypotheses

The hypotheses presented in this Chapter can be summarised as:

- The many definitions of a system are formulations of problem statements.
- The process of adjusting the area of interest in the real world to fit the cognitive filter introduces unnecessary complexity, and recognising that excessive complexity is a symptom of an underlying problem within the

foundation of the current paradigm (Chapter 3); what needs to be done is to adjust the cognitive filter to view the area of interest and create an appropriate set of simple systems.

18.12.2 The insights

The insights discussed in this Chapter can be summarised as:

- Systems cannot be totally understood from within the system even if viewed from a number of internal viewpoints.
- Several orthogonal views from an external (higher hierarchical level) perspective are also needed to completely understand a system.
- There's no such thing as a system per se! There's a map of some area of interest (aspect of the world) enclosed in a boundary that we call a system for convenience (Kline's Sysrep).
- Systems exist within hierarchies of containing systems.
- The area of interest cannot be separated from the real world.
- Defining the correct internal and external boundaries for the system and its components is critical.
- Complexity is everywhere but can be abstracted out for specific purposes, namely the way to deal with complexity is by changing the cognitive filter to a Simplicity paradigm.
- Within a given area of interest, different people may draw different system boundaries for different purposes at the same time.
- System boundaries are dynamic, not static, and depend on the prerogative of the situation.
- Attempts to establish and use a single static system boundary for all purposes results in artificial complexity.
- We do use Simplicity in a few instances, but it is not the current mainstream paradigm.
- "Complicatability" is the prerogative of the systems engineer.

18.13 Areas for future research

There are a number of areas for future research based on this Chapter including the following:

- **Hypothesis testing.** This Chapter has stated hypotheses based on the recognition that "excessive complexity is a symptom of an underlying problem within the foundation of the current paradigm" (Chapter 3) and personal experience, and extrapolated on possibilities, perceptions, outcomes and thoughts the hypotheses have generated. Future research needs to be undertaken to test the hypotheses and investigate the possibilities discussed in this Chapter.

- **Determine the Simplicity process.** Simplicity is out there in a few situations. The space station docking scenario and the LuZ SEGS-1 case study have shown that Simplicity does work. Future research needs to be undertaken to determine the nature of the process or processes for simplification and its limitations if any, so that Simplicity can become more widespread. We need to learn how to view things differently, namely adjust our cognitive filters for Simplicity not Complexity.
- **Determine the minimum number of views necessary.** Future research needs to be undertaken to determine the minimum number and types of internal and external views necessary for a complete understanding of an area of interest.

2005

19 Process architecting

This Chapter examines the system from the perspective of the work done in the development of systems. This work is currently split between three independent and apparently overlapping organisational roles of systems architecting, systems engineering, and project management, which interdependently produce a product to (the correct) specifications within the constraints of resources, budget and schedule. This Chapter first identifies a reason for the overlapping roles. The Chapter then attempts to resolve the difficulties in defining the roles of systems engineering, systems architecting, and project management, and the difficulty in defining the body of knowledge for systems engineering by identifying a gap in the functions performed by the three organisational roles, when viewed from the perspective of planning and implementing the development of a system. This gap, which when filled by the newly defined role of process architecting, has the potential to bring some order into the current chaos and resolve many if not all of the current difficulties.

There have been many discussions in the literature about the overlapping of, and differences in, the roles of systems engineering, systems architecting, and project management (Chapters 2 and 15), as well as the depth of speciality knowledge required for each of the three roles in the development of systems discussed in Chapter 7. For example, according to Roe the knowledge and skills of systems engineers are the same as those of project management in the areas of management expertise, technical breadth and technical depth (Roe, 1995). Roe adds that the difference in application is that the system engineer has more technical breadth, while the project manager has more management expertise. Bottomly et al. studied the roles of the systems engineer and the project manager and identified 185 activities and their competencies (experience and knowledge) (Bottomly, et al., 1998). Their findings included:

- No competency was assessed as being purely the province of systems engineering.

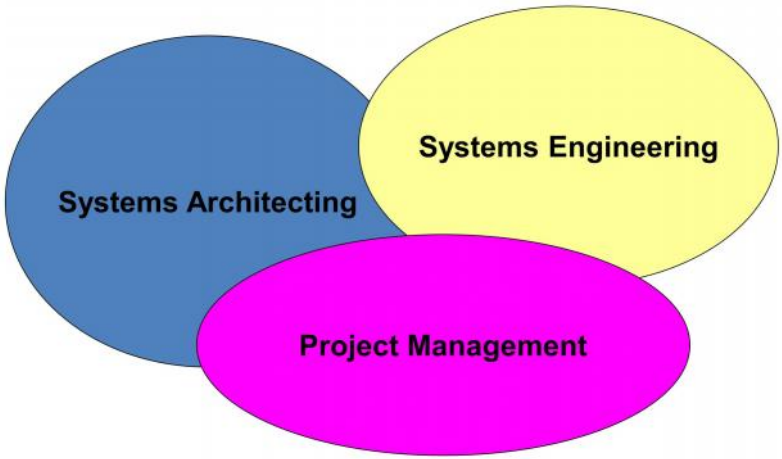


Figure 19-1 Overlapping organizational roles in the development of systems

- There is no sharp division between the two disciplines (systems engineer and the project manager) even at the level of individuals.

In these discussions, the situation tends to be represented by overlapping shapes as shown in Figure 19-1.

In addition the difficulty in defining the SEBoK has also been discussed in several places (Chapter 12). As a result of these difficulties, it has been difficult to separate the content of postgraduate courses in systems engineering, systems architecting and project management, at least in the Systems Engineering and Evaluation Centre (SEEC) at the University of South Australia (UniSA), so the current courses contain some overlapping content (although treated from different perspectives).

This Chapter first identifies a reason for the overlapping roles and then attempts to resolve the difficulties in defining the organisational roles and their bodies of knowledge. It does this by looking at the boundaries of the functions performed by the three organisational roles when viewed from the perspective of planning and implementing the development of a system and identifying a gap. When this gap is filled by a new defined role of process architecting, the four interdependent organisational roles have the potential to bring some order into the current chaos and resolve many if not all of the current difficulties.

19.1 The three current organisational functions in the development of systems

The task of developing systems is currently split between the three interdependent organisational functions of systems architecting, systems engineer-

ing, and project management which interdependently produce a product to (the correct) specifications within the constraints of resources, budget and schedule. These functions are performed by people with various roles. For the purposes of this discussion, consider the following terminology:

- **Roles** – the title or job description of a person in an organisation. These are variations of various job titles such as System Architect, System Engineer, and Project Manager.
- **Functions** – activities performed by a role in an organisation. Some functions performed by a role in one organisation may be performed by a different role in another organisation.

Now consider the three interdependent organisational functions of systems architecting, systems engineering, and project management.

19.1.1 Systems architecting

Systems architecting is defined as "*the art and science of creating and building complex systems. That part of the systems development most concerned with scoping, structuring, and certification*" (Maier and Rechtin, 2000). The function of the systems architect is to apply architectural methods analogous to those used in civil works. This function is concerned with meeting the overall client needs, directing the high-level design, focussing on keeping the interfaces between contractors manageable, and working for the client to ensure that the resulting system satisfies the client's expectations, even if the expectations are not clearly articulated.

19.1.2 Systems engineering

A number of definitions were provided in Table 12-1, which Chapter 18 postulated were all statements of problems.

19.1.3 Project management

As mentioned in Chapter 2 project management is defined as "*the planning, organizing, directing, and controlling of company resources (i.e. money, materials, time and people) for a relatively short-term objective. It is established to accomplish a set of specific goals and objectives by utilizing a fluid, systems approach to management by having functional personnel (the traditional line-staff hierarchy) assigned to a specific project (the horizontal hierarchy)*" (Kezsbom, et al., 1989).

19.2 Mapping the three roles

Given the overlapping nature, and commonality, of the functions performed by the three roles, no wonder it has been difficult to separate them. For example, Chapter 2 noted that for any phase in the SDLC, the optimal cost to

perform the phase is the "right mix" of planning and doing, but failed to separate out activities unique to the functions of systems engineering, since all the activities identified in that research as pertaining to the functions of systems engineering overlapped those of the project management functions. However, that was research was from the perspective of mapping job descriptions onto planning and implementing the development of a system.

19.3 An alternative perspective

Change the viewpoint and consider the situation from the perspective of planning and implementing the development of a system. While planning and implementing are common functions to all three roles in all organisations, the degree of planning and implementing is different at different points in the SDLC (in general, there tends to be more planning in the early phases of the SDLC and more implementation in the latter part of the SDLC). Moreover there is also a distinction between the attributes of the product being produced and the process that produces it. Thus all the functions involved in the development of systems can be mapped into the quadrants of a rectangle in which:

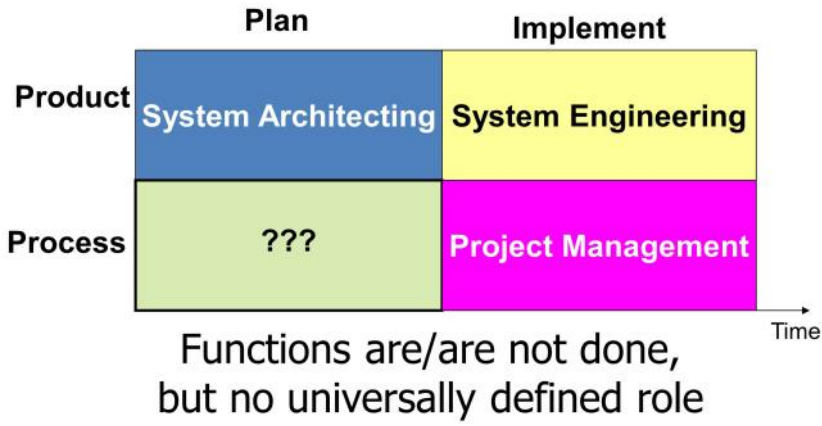


Figure 19-2 Mapping organisational functions

- the continua of planning and implementing are at the bottom showing the change in focus from planning to implementing over time (the SDLC on the horizontal axis), and
- the product or process attributes of the final system are at the side (vertical axis).
- When the three organisational functions are mapped in this way the result is as shown in Figure 19-2 in which:
  - Product-planning maps onto the functions of systems architecting.

- Product-implementation maps onto the functions of systems engineering.
- Process-implementation maps onto the functions of process engineering or project management.

19.4 Introducing the role of process architect

Figure 19-2 appears to show that the activities (functions) of process-planning are not, in general, performed by any of the three roles. This is not always so, in practice the functions sometimes tend to be incorporated in project management or in the other two roles in an ad-hoc and sometimes overlapping manner. As such, the tasks involved in project development during the planning stage of a project tend to be difficult to distinguish from those performed in project management and hence difficult to explain to students. If, for the purposes of this discussion, the functions of project management can be constrained to the process-implementation quadrant, it can be seen that the area of process-planning in the development of systems is not covered in Figure, namely a gap has been identified. This Chapter proposes that the gap be filled by a defining a fourth organisational role to:

- institutionalise and organise the process-planning functions;
- help remove the overlaps between the three current organisational roles;
- resolve some of the difficulties in teaching the roles; and hence
- clarify the bodies of knowledge for each of the roles.

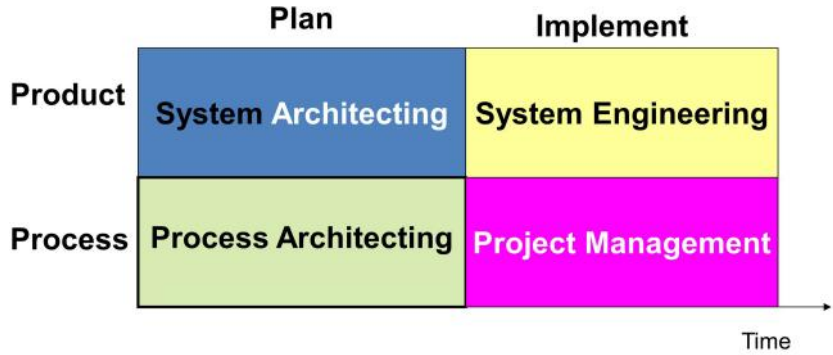


Figure 19-3 The Roles Rectangle

This Chapter also proposes that the name of the fourth role be Process Architect as shown in the Roles Rectangle of Figure 19-3 which can be used to show how the four organisational roles reinforce each other as follows:

- **System architecting** - responsible for the architecture of the system that will be produced;

- **System engineering** - responsible for the technical implementation of the design;
- **Process architecting** - responsible for the process that will produce the system; and
- **Project management** - responsible for managing the implementation of the process.

The difference between process architecting and project management may now be summed up as follows. Project management manages the implementation of the specific goals and objectives of the project (Kezsbom, et al., 1989) described by the WBS, and the Gantt and PERT charts, while process architecting is the creation of the initial (high-level) version of the WBS, the Gantt and the PERT charts used by the project manager.

In general it can also be said that the process and system architecting functions cover more than a single project because both product and development process systems must be architected to be compatible with their adjacent systems (Chapter 11), while the system engineering and project management functions tend to be limited to a single project since they are (inwardly) focussed on the completion of their systems.

19.5 The role of process architecting

The functions performed by the process architecting role are not new, they are being, or should be, performed in organisations. For example, Sheard discussed many of these functions (Sheard, 2003), and in a Case Study of a requirements elicitation and elucidation project performed in 1989, Kasser and Mirchandani stated *"During the course of this task, we thought of the systems engineer as the project expert who ensures that the process is optimally planned and implemented during the course of the project lifecycle (Kasser and Schermerhorn, 1994a). Thus not only did we perform the organizational role of systems engineering in determining the requirements for the MCSS, we also performed the organizational role of systems architecting (Maier and Rechtin, 2000) in developing the candidate architectures for the MCSS, and the role of process architecting in developing the transition plan*" (Kasser and Mirchandani, 2005).

This Chapter uses the Roles Rectangle to associate the functions of process architecting into the role of Process Architect. The following subset of the functions performed by the role of the process architecting are described in this section.

- Process design
- Process improvement
- Process change agent
- Ensuring standards and CMM compliance

- Keeper of the flame
- Business process reengineering
- Workflow analysis

19.5.1 Process design

The major organisational function of process architecting is to design, set up, and continuously optimise, the process for the development of the specific system being produced by the specific organisation over the specific time period of the SDLC to optimise productivity. ISO 15288 provides a useful list of processes for tailoring to the specific needs of a development organisation (Arnold, 2002). Moreover, since a process is in itself a system the process architecting function will use the systems engineering approach to produce the process. As NASA's systems engineering handbook stated *"Total Quality Management (TQM) is the application of systems engineering to the work environment"* (NASA, 1992b). This is not so surprising because many of the tools used for TQM are the same as for systems engineering, but with different names as NASA's systems engineering handbook stated *"Statistical process control is akin to the use of technical performance and earned value measurements"* (NASA, 1992b). Once the initial development process is designed it is turned over to the systems engineering and project management functions to construct the system.

Now there are a range of methodologies for use in the development of systems (Avison and Fitzgerald, 2003), the traditional waterfall methodology being only one of them. Each methodology fits specific scenarios; however the real world tends to be more complex than the scenarios taught in the classroom. *"Real world problems do not respect the boundaries of established academic disciplines, nor indeed the traditional boundaries of engineering"* (O'Reilly, 2004). Thus the optimal development process:

- will probably not be a straight-forward unmodified out-of-the text book methodology;
- is as important to the success of a product development as is the optimal architecture of the product; and
- is a multi-phased time-ordered sequence of activities (Chapter 13) with constraints on the start dates for each activity.

The WBS for the process looks like a hierarchical system-subsystem view of the product. However, little attention seems to have been paid to architecting optimal development processes and consequently, an untailored process may not be optimal in a specific situation. This organisational function of process architecting is to identify the best methodology for the situation and then tailor that methodology to create an optimal process for the situation. To make the function more complex, the best methodology and optimal implementation may be different at different phases in the SDLC or

the situation may be such that there is no one optimal methodology and parts of several methodologies may be to be assembled into the methodology for the project (Chapter 13). The methodology must be tailored to the situation, not the other way around. Once the methodology is chosen, the process for implementing the methodology must be developed. The choices faced by the process architecting function include:

- **Choice of lifecycle** – such as the traditional requirement driven methodology or a capability driven methodology.
- **Choice of methodology** – such as (which) soft-systems, functional, object-oriented, waterfall, rapid, spiral, cataract etc.
- **Choice of process for implementing the methodology**, milestone process-products and the checkpoints within the process. The process must be scaled to the size of the project. Sometimes this may require combining activities or products, e.g. combining the operations concept with the systems requirements documents for small projects, or even choosing to produce milestone documents in the form of PowerPoint presentations instead of text mode documents.
- **Build – buy decisions**. The decision to build or buy components of the product affects the development process as well as the product architecture. This decision must be made after considering its implications on both the product system and development process.

19.5.2 Process improvement

The process architecting function is responsible for monitoring and improving the process. Process improvement should be performed by persons outside the process, but intimately acquainted with it (Kasser, 1995). The outsider is important because apart from the different perspectives they bring, those involved in the process generally are too busy to spend any time improving the process, and if they are not too busy, they generally are not open to change. The process architecting function is that of the Quality Guru as far as the development process is concerned.

Process improvement however, is more than just applying process standards. Process standards document observed activities that have led to successes. The Standards need to be tailored to suit the specific project in the specific organisation at the specific time. Process architects need to know when to go by the book and when to write the book since the literature on excellence has little if anything to say about the CMM (Peters and Waterman, 1982; Peters and Austin, 1985; Rodgers, et al., 1993). The literature discusses the need for knowledgeable people to get things done. When going from chaos to order an improvement will usually be observed. That takes you to CMM Level 5 or to ISO 9000 compliance, but what then? Standing at the bottom of the process improvement mountain you only see the foothills leading to the plateau at Level 5 as shown in Figure 19-4. Level 1 is

categorised by having success achieved by heroes. Levels 2-5 discourage heroes and focus on orderly processes. It will take heroes working within the organised organisation structure to effect further improvements beyond Level 5 and improve your competitive edge. Companies don't want employees who can follow rules; they want people who can make the rules (Hammer and Champy, 1993) page 70). Winning (world-class) organisations need to focus on individual excellence and reward individuals for their achievements and the risks that they are willing to take (Harrington, 2000).

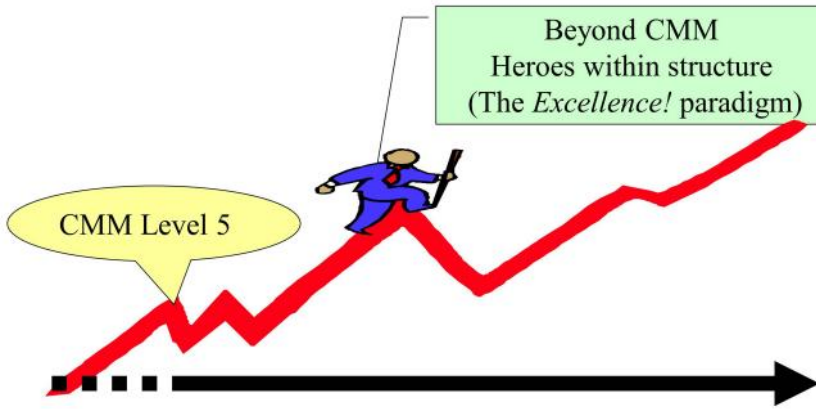


Figure 19-4 The process improvement mountain

19.5.3 Process change agent

Changes to the process may become necessary during the implementation phase of the SDLC. These changes arise for various reasons including the necessity to reduce time to market, major changes in the product specifications, and changes in the resources available for the development process (new resources become available or reserved resources become unavailable, or a combination of both). In this scenario, the process architecting function redesigns the methodology for producing the product, the analogy in the product arena is responding to changes in product requirements. This function requires close cooperation with the other three organisational functions.

19.5.4 Ensuring Standards and CMM compliance

The process architecting function has the organisational responsibility of ensuring compliance to the various standards such as ISO 15288 (Arnold, 2002) and the CMM appropriate to the development, or mandated by the customer. Should the organisation not be complaint, and need to become so, the process architecting function designs the compliant process and the

transition approach (Chapter 4 and 6).

19.5.5 Keeper of the flame (process)

This is the function of responsibility for knowledge management and learning within the organisation for process related matters including,

- management of process related lessons learned on various projects by documenting them and making them available for subsequent projects;
- defining process-related training needs;
- a knowledge of applicable government and other regulations and standards to which the process must conform; and
- ensuring that all appropriate supply chain requirements[79] are levelled on the product design (using the object-oriented concept of inheritance). In the Defence industries, an example of a supply chain requirement is the need to ensure that pre-assembled equipment designed for use on a submarine will fit through the entry hatches at installation time. In the commercial arena, it may have to do with storage requirements, packaging requirements, third-party installation requirements, ensuring that shipping containers will fit on the vehicle, etc.

19.5.6 Business process reengineering

This is a process architecting function. In fact the process architecting function should be responsible for designing the structure of the organisation in which the development system exists in accordance with the ISO 15288 standard (Arnold, 2002).

19.5.7 Workflow analysis

This is a process architecting function. The goal is to analyse the workflows identify the optimal methodology and design the best process for the situation.

19.6 Interdependence in the Roles Rectangle

The two dimensions of the Roles Rectangle provide a simplified representation of the four organisation functions (not the roles with similar names) from the perspective of planning and implementing the product and the development process producing the product. Each architecting function requires knowledge of the functions of both implementation functions since for example:

[79] A category of missing requirements in many instances.

- **Product** - there is little pointing designing a product that cannot be produced either because the specifications are unachievable (i.e., requirements to travel faster than the speed of light are not achievable with today's technology).
- **Process**- there is little point in setting schedules that are not feasible due to lack of resources, or time.

In addition, each implementation function also uses some planning. Moreover, each quadrant in the Roles Rectangle contains the functions performed by the speciality disciplines. For example, Eisner describes 38 speciality disciplines in systems engineering alone (Eisner, 1988). Some of these disciplines are present in the other three quadrants. For example risk management is a function that has attributes in all four quadrants. The Roles Rectangle can be used to show that risk management should be applied in all four quadrants, and the nature of risk management in each quadrant can be identified using planning and implementing as a starting viewpoint. The bodies of knowledge in courses teaching risk management can be organised according to the quadrants of the Roles Rectangle.

19.7 Mapping the organisational functions to the organisational roles

If there was a one-to-one mapping of the roles to the functions, then there would be little discussion as to the differences between roles of the systems engineer and the roles of the project manager. All the functions in the systems engineering quadrant would be performed by the systems engineer, and all the functions in the project management quadrant would be performed by the project manager. It is when the boundaries of a role, as defined by the job description, contain functions in another quadrant that discussions arise.

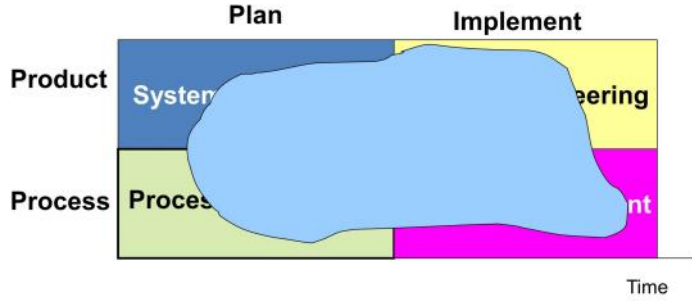


Figure 19-5 Role of systems engineer in one organisation

The work in developing systems is interdisciplinary. It incorporates a large number of engineering, management, and other functions that have to

be performed (e.g. requirements management, design, decision making, problem solving, validation and verification, test and evaluation, risk management, reliability, and logistics, process design and improvement, etc.) (Watts and Mar, 1997). In small projects, one person might perform all the functions. On larger projects, the functions tend to be grouped (slightly) differently in different organisations in different jobs that are not exactly aligned with the organisational functions. Thus a systems engineer's job does not exactly align with the functions of systems engineering. As both Roe and Sheard noted, a systems engineer can perform some systems engineering functions and also perform some project management functions (Roe, 1995; Sheard, 1996). They can also perform architecting functions, yet the job description is "Systems Engineer". However, in a different organisation, the partition of work into different jobs is also not exactly aligned with the organisational functions but in a different way. This means that in different organisations, the partition of work between the jobs of Systems Engineer, Project Manager, and Systems Architect will probably be different. Thus the same person known as a "Systems Engineer" or an "Engineering Specialist" might perform a different subset of systems engineering, project management, and systems architecting when moving from one organisation to another as shown in Figure 19-5 and Figure 19-6, by the different overlap of their roles (the activities they do or functions they perform) into the four (defined by name) activity rectangles but the same job names are used, e.g. "Systems Engineer" and "Engineering Specialist". This situation means that in any one organisation, in general, the roles performed by the jobs of systems engineer, systems architect and project manager while they do not map directly into their corresponding function rectangles, also do not overlap each other's roles (unless there is a turf war in progress). The functions only overlap job roles when compared across different organisations.

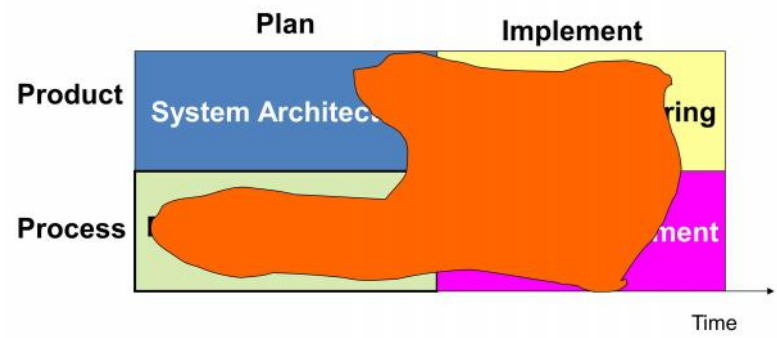


Figure 19-6 Role of system engineer in another organisation

19.8 Traits for a process-architect

The traits for a process architect can be identified by building on those for the systems engineer. Arthur D. Hall provided the specifications or traits for an "ideal systems engineer" (Chapter 2), other skills and knowledge can be identified such as those discussed by Watts and Mar (Watts and Mar, 1997). The specifications or traits for process-architecting are basically the same but in the methodology and process domain rather than in the application or product domain. Thus an understanding of the methodologies described by Avison and Fitzgerald (Avison and Fitzgerald, 2003) and when each should be used, or similar knowledge, would be a fundamental requirement. Moreover, as stated above, an effective process architect must have some understanding of, and experience in, the functions performed in the other quadrants of the Roles Rectangle.

19.9 Summary

The job description (role) of a systems engineer, in general does not map directly into the functions of system engineering in the work of developing systems. Similarly the job descriptions (roles) for project managers and systems architect do into map directly into the functions of project management and systems architecting. Since, in general, the jobs (roles) overlap when compared across different organisations, because the boundaries of the job functions allocated to the jobs are different in different organisations, confusion has arisen as to the differences between the responsibilities of the job descriptions and the functions.

The role and function of process architecting has been introduced as a way to resolve these difficulties in differentiating between the roles of the systems engineer, system architect, and project manager, and the functions of systems engineering, system architecting, and project management in the development of systems across organisations. The use of the Roles Rectangle which views the development of systems from the perspective of the difference in the degree of planning and implementing over the SDLC and a distinction between the product being produced, and the development process producing the product, portrays the roles and functions in a simplistic manner which clarifies the roles at a conceptual high level. However, it must not be forgotten that the functions performed in the speciality disciplines are embodied in each of the roles. Yet when discussing the speciality disciplines with respect to the Roles Rectangle, the aspect of the discipline associated with each rectangle can be readily identified.

19.10 Conclusions

The introduction of the Roles Rectangle makes it easier to define the organi-

sational roles of the systems engineer, system architect, and project manager, and the functions of systems engineering, systems architecting, and project management, as employed in the development of systems by adding the fourth role of process architect and identifying the functions of process architecting. The use of the Roles Rectangle assists in:

- teaching about the arrangement of work in the development of systems,
- understanding the organisational roles by which work is partitioned, and
- the mapping of job descriptions into those roles.

Consequently it provides a way to partition knowledge across a number of courses on systems engineering and project management with less overlap than in the current syllabus.

Lastly, the role of systems engineering in the development of systems is more clearly and tightly scoped. This sets a boundary for the SEBoK focussed on the product-implementation phases of the SDLC, which can be used in a certification program for systems engineers (Chapter 7).

2007

20 Eight deadly defects in systems engineering and how to fix them

There is a growing trend towards the adoption of systems engineering in the belief that systems engineering has the ability to perform the acquisition and maintenance of the systems that underpin our society. Examples include:

- The health care industry in the US has failed to adopt systems-engineering tools and new technologies that could significantly improve the safety and quality of medical products and services while also lowering costs (Reid, et al., 2005).
- The US DOD has mandated the use of a "*robust systems engineering approach*," although with little definition of what this means (Wynne, 2004) as quoted by (Honour and Valerdi, 2006).

Any organization wanting to adopt or improve systems engineering needs to be aware that research into the nature of systems engineering has shown that when viewed from an external perspective, systems engineering as currently practiced, contains a number of defects (Kasser, 2006). These defects inflate the cost of acquiring and maintaining systems. Fixing these defects should reduce costs and may mitigate some of the need to develop new tools and techniques to manage complex systems. While experienced systems engineers may be aware of the defects, the fact that the defects are current and not historic implies that in general systems engineers are not aware of them. The eight defects in systems engineering discussed in this Chapter are:

1. The selection of independent alternative solutions;
2. The misuse of the V diagram;
3. The lack of a standard process for planning a project;
4. The abandonment of the Waterfall model;
5. Unanswered and unasked questions;
6. Lack of a metric for the goodness of requirements;
7. A focus on technological solutions;
8. The need to focus on people as well as process.

While some systems engineers perform a version of systems engineering which does not include these defects, they tend to be the exception rather than the rule. Consider some aspects of these defects in turn, and how fixing them would lower the cost of doing work.

20.1 The selection of independent alternative solutions

Text book systems engineering generally discusses the selection of alternatives with the implicit assumption that the selected alternative is also the optimal one, e.g. (Blanchard and Fabrycky, 2006) page 41). Consider the following scenario. The systems engineering process has identified three alternative candidate solutions (A, B and C). "C" gets the highest score (even after the sensitivity analysis to make sure the weightings were reasonable). "C" gets selected, but which of the candidate solutions is the optimal solution? The answer is possibly none of them is optimal because each of the different design teams will generally have different strengths and weaknesses so different parts of different solutions will be better or worse than the corresponding part of their counterpart systems. The optimal system may be a combination of the best parts of all of them yet we do not get to examine this combination. However, fixing this defect may require a change to the acquisition contract paradigm.

20.2 The misuse of the V diagram

The V diagram is often described as a depiction of the systems engineering process. Practitioners however tend to forget or are unaware that it is a three dimensional model and in its two-dimensional representation it is only an overview of some of the aspects of the project cycle relating development to T&E at the various phases of the SDLC while abstracting out all other information. The V diagram was initially introduced into both software and systems engineering as a project management tool.

A literature search found the first mention of the V diagram (Rook, 1986) where it was introduced as a software project management tool illustrating the concept of verification the process-products at established milestones. The original figure extracted and shown in Figure 20-1 was captioned "*the stages in software development confidence*". It was drawn to show that the intermediate process products produced at each phase of the software development were to be verified against previous baselines before starting work on the subsequent phase. The V diagram seems to have been introduced to the systems engineering community (Forsberg and Mooz, 1991) also as a project management tool. Forsberg and Mooz and Rook state that the simplistic view of the product development cycle is not to be interpreted as a waterfall namely that each phase is to be completed before the next begins. They agree that explanatory work on subsequent phases is

often required before a phase is complete and there is a third dimension to the model. Forsberg and Mooz include a representation of that third dimension in their paper and one of their figures, extracted from their paper is shown in Figure 20-2.

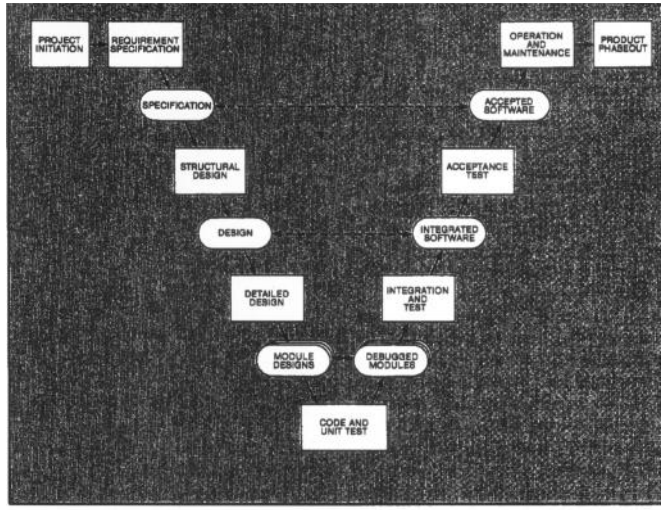


Fig. 3 The stages in software development confidence

Figure 20-1 The V diagram for software development (Rook, 1986)

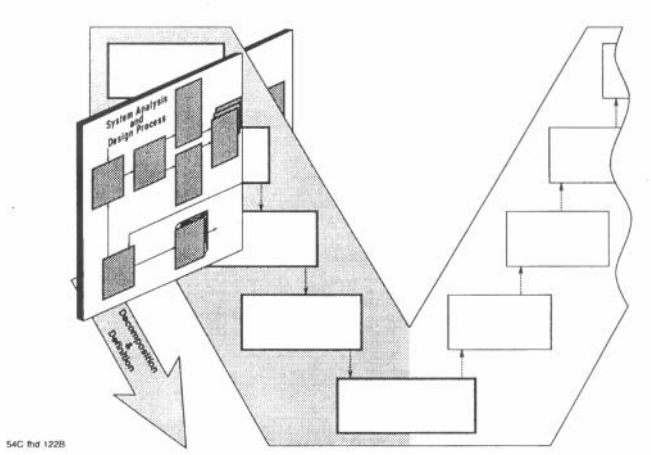


Exhibit 8—Application of the System Analysis and Design Process to the Technical Aspect of the Project Cycle

Figure 20-2 The three dimensions to the V diagram (Forsberg and Mooz, 1991)

In addition, when the V diagram is used in a simplistic manner to depict the relationship between development and T&E there seems to be no place in the diagram for the 'prevention of defects'[80]. While the development team implements the system, the test team is busy planning the tests. A definition of a successful test is one that finds defects[81] (Myers, 1979). This is because if no defects are found, the result is ambiguous, because either there are no defects or the testing was not good enough to detect any defects. The lack of prevention escalates costs. Deming wrote "*Quality comes not from inspection, but from improvement of the production process*" (Deming, 1986) page 29). He also wrote "*Defects are not free. Somebody makes them, and gets paid for making them*" (Deming, 1986) page 11). If the test team can identify defects to test for, why can't they hold a workshop or other type of meeting to sensitise the development team to those defects and hence prevent them from being built into the system? Such workshops in postgraduate courses at UMUC and UniSA have sensitised students to the problems caused by poorly written requirements (Kasser, et al., 2003). The development and test teams need to work interdependently not independently (Kasser, 1995).

The V diagram needs to be used in context in its full three dimensional version (Forsberg, et al., 2000) with the explicit addition of prevention.

On the subject of prevention, the systems engineering process could also benefit from the adoption of poka-yokes (procedures that prevent mistakes) (Chase and Stewart, 1994) as quoted by (Chase, et al., 1998) page 155).

20.3 The lack of a standard process for planning a project

Systems engineering has often been described as a process, e.g. (MIL-STD-499B, 1992). However, it lacks a standard process element for the planning phase of a task. A typical approach is shown in Figure 20-3; a suggested improved process based on experience is shown in Figure 20-4. The contribution of this process is two important elements that are not generally performed, namely:

- Identification and application of lessons learned from prior projects,
- Negotiation of objectives and resources.

After the task begins the process architect determines the objectives and available resources. Sometimes they are provided, other times they

[80] This is dicussed in section 26.3.

[81] As opposed to the goal of the system development team to produce a defect free system.

have to be identified. The process architect should ask the following questions (Chapter 13):

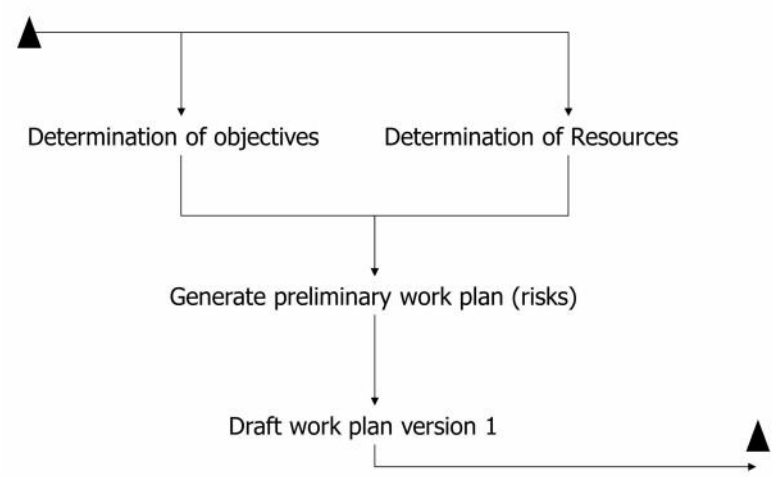


Figure 20-3 Typical approach to planning a project

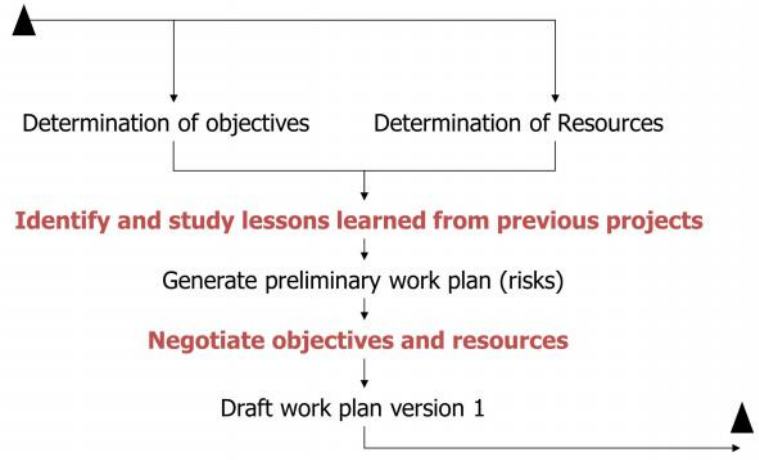


Figure 20-4 Process for starting a task

1. Has anyone done this task or a similar one before?
2. Did they succeed or fail?
3. Why?
4. What is the difference between the other task(s) and this one that might change those results?

Obtaining answers to these questions, performing the analysis, and presenting the result, requires access to the organization's lessons learned da-

tabase. Once access is provided, the first action is to determine if anyone has faced a similar task and identify the lessons learned from those tasks. The process architect identifies what worked, what didn't work in the previous situations; compares the situations to the current one and determines if those factors apply, and what effect they may have. This step of the process may be thought of as prevention, pattern matching, risk management or even inheritance in its object-oriented sense. This step is critical since it can prevent mistakes from being made, and wrong approaches from being taken. Yet process methodologies such as PRINCE2 (Bentley, 1997) generally require the lessons learned to be documented at the end of a process but do not require that they be reviewed at the start of the next project. Project lessons learned documents seem to be write-only memories except in CMM Level 5 organisations!

The outputs of this stage of the process are the initial risk management plan for process related and some product related risks together with the draft schedule and WBS. The next phase of the process is to negotiate the objectives and resources. The draft WBS and schedule is adjusted to meet the needs of the situation. This process, incorporates prevention of defects (at least some known ones) by definition, hence reduces the cost of doing work. Early identification of inadequate schedule time or resources allows the project manager to attempt to deal with the situation in a proactive manner, rather than a reactive manner in the implementation phase of the project. Once any adjustments have been made as a result of the negotiations, the process architect turns the initial version of the work plan over to the project manager and the task begins.

If realistic schedules and objectives are set, the project manager is able to plan ahead, anticipate and implement changes, so schedules and budget goals are met. As a consequence, the project receives very little senior management visibility. All goes reasonably well, and in the main, senior management in general, does not realize or recognize the achievements of the process architect and project manager[82] (Kasser, 1995) page 4). If unrealistic schedules are set, or insufficient resources are allocated to the project, the project will be doomed, but at least everyone will know why ahead of time! This situation manifests itself in the John Wayne style of reactive management, continually fighting crises, leading to high visibility (Kasser, 1995) page 135). All the problems are visible to senior management, who tend to reward the project manager for saving the project, sometimes from the same problems that the project manager introduced. As Deming wrote: "***Heard in a Seminar:*** *One gets a good rating for fighting a fire. The result is visible; can*

[82]There weren't any problems, so obviously cost and schedule were underestimated.

be quantified. If you do it right the first time, you are invisible. You satisfied the requirements. That is your job. Mess it up, and correct it later, you become a hero" (Deming, 1986) page 107). Thus if you want to be promoted, your approach should be to build failure and recovery into the project. Instead of heading problems off, anticipate them, but let them happen. Only apply corrective measures after the making the problems visible to upper management. If you implement a project in this manner, it will make you a hero at the expense of the organization.

20.4 The abandonment of the waterfall model

The waterfall model (Royce, 1970) has been deemed a failure because of the effect of changes and the need for iterative development cycles. However, the original waterfall model shown in Figure 13-1 works very well when the requirements are known and don't change during the development time (Chapter 13). However, in general development takes so long that the requirements change. This condition has been depicted in the chaotic view shown in Figure 13-2. However with some configuration control, the model can be drawn as shown in Figure 13-4. The spiral model (Boehm, 1988) is basically the waterfall model with risk management emphasised. However, the spiral does not go far enough, the functionally provided by the system produced by development needs to converge on the changing needs of the user which makes configuration management and control of change critical components of the development process.

A series of mini-waterfalls or cataracts can provide that evolutionary convergence. This model is shown in Figure 13-5 as the Cataract model (Chapter 13) where each Build contains a short duration waterfall under the control of a CCB as shown in Figure 8-1. The Cataract model is not iterative or sequential; it is a multi-phased time-ordered parallel-processing recursive paradigm that can be used to control the development of both systems including so-called systems of systems in a more cost-effective way than today's other development methodologies.

20.5 Unanswered and unasked questions

20.5.1 Unanswered questions

There are two critical questions that cannot be answered accurately in today's systems engineering development paradigm at any time during the SDLC (Chapter 5 and 6). These unanswerable questions posed by the supplier to the customer are:

- What is the exact percentage of completeness of the system under construction?

- What is the probability of successful completion within budget and according to schedule?

Given that some systems development can take many years, and the customer is paying for work in process, this is critical information when the system under development is needed to work in a family of other systems to provide some needed capability at some future time.

20.5.2 Unasked questions

There are several questions that are not asked in today's paradigm because the information has not been assembled in such a manner as to make people think about asking them. These questions that are not asked today at System Requirements Review time have been identified by the object-oriented approach (Chapter 17) and include the following:

- **Have the high priority requirements been assigned to early Builds?** This allocation ensures that if funds are cut during the development time, the customer has a high probability of receiving the most important functionality.
- **What is an appropriate Risk Profile for the type of system to be realised and is that the same profile as the instance being realised?** Many systems are repeats of previous systems with modifications. Risk Profiles may be able to help identify the technological uncertainty associated with the project (Shenhar and Bonen, 1997). Comparisons with similar projects may be able to identify potential problems and allow proactive risk management.
- Are requirements with the following pair of properties really needed (at SRR time)?
    - High cost, high risk.
    - Low priority, high cost.
    - Low priority, high risk.

Are the features driving these requirements really needed? Think of the cost savings if these requirements are eliminated before work takes place implementing them. Developing ways of answering these questions should lower the cost of doing work. Tran and Kasser have performed research into developing ways of presenting decision makers with the information to allow them to more easily pose and answer these questions (Tran and Kasser, 2005).

20.6 The lack of a metric for the goodness of requirements

"*It has been known since as early as the 1950s that addressing requirements issues improves the chance of systems development success*" (Buren and

Cook, 1998). There has been a lot of research into building the right system and doing requirements better (Glass, 1992). Much of that research has focused on how to state the requirements in the form of a specification once they have been obtained, using a RTM, and the tools that incorporate a RTM. However, recognition that the current paradigm produces poorly written requirements has been documented at least as early as 1993 (Hooks, 1993) and various approaches have been since proposed to alleviate the situation without much success[83]. For example:

- Jacobs states that a 1997 analysis of the software development process performed at Ericsson identified "missing understanding of customer needs" as the main obstacle for decreasing fault density and lead-time (Jacobs, 1999). Related findings were aggregated under the heading "no common understanding of 'what to do'". The counter measures to overcome these problems focused on testing the quality of the requirements rather than producing good requirements. There was no proposal on how to get clear requirements, nor was there a clear understanding of what a clear requirement was.
- Goldsmith states that the process of "*defining business requirements is the most important and poorest performed part of system development*" (Goldsmith, 2004).

Thus, there is a consensus that good requirements are critical to the success of a project. System engineers have focused on generating requirements to ensure that the as-built system is fit for its intended purpose. Requirements Engineering is a discipline that is evolving from its traditional role as a mere front-end to the systems lifecycle towards a central focus of change management in system-intensive organizations (Jarke, 1996). For example:

- The definition of requirements engineering as "*the science and discipline concerned with analysing and documenting requirements*" (Dorfman and Thayer, 1990).
- The definition of requirements engineering as "the systematic process of eliciting, understanding, analysing, documenting and managing requirements" (Kotonya and Summerville, 2000).

However, there is no universally accepted metric for the goodness of requirements either individually or as a set in a specification (Kasser, et al., 2006). We need to develop one and a number of ways of developing such metrics were proposed by Kasser et al. (Kasser, et al., 2006).

[83] Because the B paradigm is inherently flawed (Chapter 28).

20.7 A focus on technological solutions

Systems engineering traditionally focuses on the technological part of the system particularly in the USA with its focus on high technology. This was recognised as early as 1959 by Goode and Machol who wrote "*the systems engineer is primarily interested in making equipment changes*" (Goode and Machol, 1959) page 130). However, when the real problem is addressed, technology alone may not provide the solution. For example the problem the executive had was "*to secure at all times, live and accurate data concerning the exact conditions of the business*" (Farnham, 1920) page 20). Yet, in the subsequent 85 years, we have not developed an accounting system which tells the decision makers what their costs really are or a management information system that provides pertinent information for making an informed decision between two alternative courses of action.

Fifty years after Farnham called for a management information system that would provide a solution to the executive's need. Stafford Beer provided a description of such a conceptual information system. Beer discussed the British War Room in the Battle of Britain and NASA's control room at the Manned Space Flight Center in Houston, Texas as close parallels (Beer, 1972) page 244). He wrote that bits and pieces of it (the system) existed. Yet in the intervening years, although the technology to build such a system has become commonplace, it still does not exist, at least in the literature. While Beer proposed a centralized control centre, today's technology allows for personal desktop portals accessing information via software agents in an integrated digital or network centric environment. In such an environment, information pertaining to the process flows in the organization, the resources available and schedules would be accessible via software agents (Chapter 8). This information exists in digital form in most organizations; it is just not readily accessible in a manner to assist the decision makers.

To me, the most successful management information system of the 20th Century did not contain a single mechanical or electronic computer. It was the command and control system employed by the Royal Air Force (RAF) in the Battle of Britain in 1940. The system contains the radar sites, the Observer Corps, the communications links and the various headquarters and operations rooms. The RAF evolved the system, the developers working together with the customer. The people were integrated with the technology. When the system entered into service it was staffed by personnel who understood the situation at the other end of the interface. Thus pilots staffed the operations room and spoke to the pilots in the squadrons. This system was developed in the late 1930's before "systems engineering" was recognised as a discipline. The developers didn't stop to discuss if it was a system, a System of Systems or a family of systems, they just developed it. So even though Germany had better radar technology, the RAF integrated

their adequate technology into a system and used it to help win the battle (Bungay, 2000).

Even though the system was successful, lessons can still be learned from two of the failings of this system which resulted in preventable downtime of parts of the system. When the parts of the system went down, a window was opened up in the air defence system which allowed entry to the enemy aircraft. The two failures were:

- The radar sites and operations rooms were dependent on externally generated electricity from the Power Grid. When the Grid connections were destroyed by enemy action, parts of the system went off-line until repairs were affected and power was restored. Standby power generators should and could have been deployed as part of the installations.
- The operations rooms were co-sited with airfields for convenience. On occasions when the airfields were bombed, the operations rooms were damaged and taken off line for short periods of time.

It should be pointed out that the effects of these failings were minor due to the tactics employed by the Luftwaffe in the battle. However, an alternative set of tactics pointed out by (Bungay, 2000) could have exploited these defects to cause much more and serious damage to the RAF infrastructure. Modern systems engineering needs to be able to develop systems in the same manner as the RAF developed this system without overlooking similar types of defects.

20.8 The need to focus on people as well as process

Systems engineering is perceived as a process[84] (Hall, 1962; MIL-STD-499B, 1992) and there is a major focus on process standards and the CMM. The contribution of effective people and the difference they can make is generally overlooked. Henry Ford wrote "*the best results can and will be brought about by individual initiative and ingenuity – by intelligent individual leadership*" (Ford and Crowther, 1922). The contribution of good people in an organisation was recognised in the systems engineering literature about 50 years ago, namely *"Management has a design and operation function, as does engineering. The design is usually done under the heading of organization. It should be noted first that the performance of a group of people is a strong function of the capabilities of the individuals and a rather weak function of the way they are organized. That is, good people to a fairly good job under almost any organization and a somewhat better one when the organi-*

[84] See Section 29.2.2.

zation is good. Poor talent does a poor job with a bad organization, but it is still a poor job no matter what the organization. Repeated reorganizations are noted in groups of individuals poorly suited to their function, though no amount of good organization will give good performance. The best architectural design fails with poor bricks and mortar. But the payoff from good organization with good people is worthwhile" (Goode and Machol, 1959) page 514).

Bungay in summarising the people in the Battle of Britain discusses the differences between Air Vice-Marshalls Keith Park and Trafford Leigh-Mallory who commanded different Fighter Groups. Bungay then continues "*What Park achieved in the Battle of Britain is in itself enough to pace him amongst the great commanders of history. But his performance in 1940 was not a one-off. In 1942 in Malta, Park took the offensive and turned Kesselring's defeat into a rout. After that, he directed the air operations that enabled Slim to expel the Japanese from Burma. He was as adept at offence as he was at defence, and, like Wellington, he never lost a battle. His record makes him today, without rival, the greatest fighter commander in the short history of air warfare"* (Bungay, 2000) page 383). In 1940 Park and Leigh-Mallory had the same processes based on (RAF tactics and doctrine), yet it was not the superiority of the RAF process to that of the Luftwaffe that made the difference[85], it was the person who made the difference[86]. One was an administrator, the other a leader!

The literature is full of advice as to how to make projects succeed; typical examples are (Rodgers, et al., 1993; Peters and Waterman, 1982; Peters and Austin, 1985; Peters, 1987; Harrington, 1995) which in general tend to ignore process and focus on people. Systems engineers focus on developing processes for organisations – namely the rules for producing products. Companies don't want employees who can follow rules; they want people who can make the rules (Hammer and Champy, 1993) page 70). Excellence is in the person not the process. This was recognised as early by Hall's specifications or traits for an "Ideal Systems Engineer" (Chapter 2). In the intervening years, process standards such as ISO 9000 and the various CMMs have proliferated. Yet the Standards do not provide metrics that can predict the failure of a project[87].

Consider the CMM. Standing at the bottom of the process improvement mountain you only see the foothills leading to the plateau at CMM Level 5 as

[85] In fact, as Bungay points out, the RAF tactics for fighter formations was inferior to that of the Luftwaffe and cost the lives of many pilots until the survivors learnt to ignore them.

[86] As another example, consider the service at your favourite restaurant. Do all table staff provide the same level of service, or are some better than others?

[87] See Section 6.7.

shown in Figure 19-4. When you get to CMM Level 5 which way do you look, back the way you came, or further up the mountain? If you look back, you see that CMM Level 1 is categorised by having success achieved by heroes. CMM Levels 2-5 discourage heroes and focus on orderly processes. However, going from chaos to order should always produce an improvement. We need to look at cost effective ways of improvement. We need to look forward up the mountain and explore what lies beyond the base camp at CMM Level 5 to continue the journey further up the process improvement mountain. Thus CMM Level 5 is only a start.

We need to research how effective people can best be used in organizations where there is order. How does the world of agile processes fit within the CMM? Once process are documented and followed, then the next step might be to treat process elements as components of a system (the whole process) and process architect agile systems to meet various needs. Research needs to be done to define the relationship between the CMM levels, effective people and agile process/systems development.

20.9 Summary

This Chapter has discussed eight of the defects in today's systems engineering paradigm. The Chapter also discussed the need for further research because fixing these defects has the potential to significantly reduce the cost of acquiring and maintaining the systems that underpin our 21st century civilization.

20.10 Conclusion

Adopting systems engineering without fixing these defects will not realise the potential of systems engineering. We need to train more effective systems engineers.

2007

21 Introducing a framework for understanding systems engineering

While the world is turning to systems engineering to solve the problems of developing and maintaining the systems underpinning our civilization, after 50 years of trying, development projects are still characterized by cost and schedule overruns as well as outright cancellations (CHAOS, 2004). Against this background,

- Systems engineering is still struggling to be recognised as an engineering discipline in an environment in which it is perceived to overlap the activities of project management.
- While universities offer degrees in systems engineering and pursue research, systems engineering still lacks a framework for research and education.
- Systems engineers can't agree on what systems engineering is (activity).
- Systems engineers can't agree on what systems engineers do (role).
- Systems engineers can't agree on a definition of systems engineering.

This situation needs to be remedied. Research has shown that one reason for the lack of agreement is that systems engineers do many and different tasks in their work and consequently have different perspectives on systems engineering (Chapter 18). In addition trying to understand systems engineering seems to be like solving a wicked problem (Rittel and Webber, 1973). Wicked problems have the following ten characteristics:

1. There is no definitive formulation of a wicked problem.
2. Wicked problems have no stopping rule.
3. Solutions to wicked problems are not true-or-false, but good-bad,
4. There is no immediate and no ultimate test of a wicked problem.
5. Every solution to a wicked problem is a "one-shot" operation"; because there is no opportunity to learn by trial-and-error, every attempt counts significantly.

6. Wicked problems do not have an enumerable (or an exhaustively describable) set of potential solutions, nor is there a well-described set of permissible options that may be incorporated into the plan.
7. Every wicked problem is essentially unique.
8. Every wicked problem can be considered to be a symptom of another problem.
9. The existence of a discrepancy representing a wicked problem can be explained in numerous ways. The choice of explanation determines the nature of the problem's resolution.
10. The planner has no right to be wrong[88].

Many of the characteristics of wicked problems also seem to manifest in the first phase of the scientific method of solving problems, namely the observation phase. The scientific method can be considered as a lifecycle consisting of:

1. Observation of the system without an understanding of the system.
2. Formulation of a hypothesis to explain the system.
3. Use of the hypothesis to predict the behaviour of the system in various situations.
4. Testing of the hypothesis to determine if the system behaves as predicted in those situations.

It is possible that wicked problems manifest themselves in the first step of the scientific method cycle. That is, the system or a part of it is under observation, but no working hypothesis to explain the system has yet been developed. For example, the state of the art of chemistry before the development of the periodic table of the elements could be considered as a wicked problem, as could the state of electrical engineering before the development of Ohm's Law.

If today's systems engineering is in a similar state, namely solving wicked problems, then a major step forward in the development of the discipline would be to apply the scientific method to the problem formulate and test hypotheses and eventually evolve a working framework for applying systems engineering in the manner of the application of the periodic table of elements in chemistry. The process of thinking about the observations, and applying the scientific method would be enlightening in itself. This Chapter takes the plunge and lists four requirements for a framework for understanding systems engineering, then documents the evolution of one such a framework by viewing systems engineering from the perspective of problem solving.

---

[88] The planner was the person facing the wicked problem in the context of the cited publication.

21.1 The need for a framework for understanding systems engineering

We need a framework because systems engineering has failed to fulfil 50 years of promises of providing solutions to the complex problems facing society. Wymore pointed out that it was necessary for systems engineering to become an engineering discipline if it was to fulfil its promises and thereby survive (Wymore, 1994). Nothing has changed in that respect since then. He also wrote that "*Systems engineering is the intellectual, academic, and professional discipline, the principal concern of which is to ensure that all requirements for bioware/hardware/software systems are satisfied throughout the lifecycles of the systems. This statement defines systems engineering as a discipline, not as a process. The currently accepted processes of systems engineering are only implementations of systems engineering*". Out of more than 50 definitions discovered in the literature discussed in Chapters 12 and 18, Wymore provided the only definition of systems engineering as a discipline.

21.2 Systems engineering as a discipline

If systems engineering is going to be recognized as a discipline we need to consider the elements that make up a discipline. One view of the elements of a discipline was provided by (Kline, 1995) page 3) who states "*a discipline possesses a specific area of study, a literature, and a working community of paid scholars and/or paid practitioners*".

Systems engineering has a working community of paid scholars and paid practitioners. However, the area of study seems to be different in each academic institution but with various degrees of commonality. This situation can be explained by the recognition that

1. systems engineering has only been in existence since the middle of the 20th century (Johnson, 1997; Jackson and Keys, 1984; Hall, 1962), and
2. as an emerging discipline, systems engineering is displaying the same characteristics as did other now established disciplines in their formative years[89].

Thus, systems engineering may be considered as being in a similar situation to the state of chemistry before the development of the periodic table of the elements, or similar to the state of electrical engineering before the

[89] It may have taken so long to recognize the situation because few if any living systems engineers were around in the formative days of chemistry and electrical engineering.

development of Ohm's Law. This is why various academic institutions focus on different areas of study but with some degree of commonality in the systems development lifecycle. Nevertheless, to be recognized as a discipline, the degree of overlap of the various areas of study in the different institutions needs to be much, much greater.

21.3 Moving towards a discipline

Assuming today's systems engineering is in a similar state to chemistry and electrical engineering in their formative years, then, as stated above, a major step forward in the development of the discipline would be to apply the scientific thinking stream of systems thinking (Richmond, 1993) to postulate a hypothesis (namely a framework for understanding systems engineering exists); identify and prototype a candidate framework, test it; modify it and eventually evolve a working framework for understanding systems engineering. The application of this framework would then pull together systems engineering in an analogous manner to the application of the periodic table of elements in chemistry.

For systems engineering to be a discipline according to Kline's definition, then the specific area of study needs to be defined. This requires some research. The hypothesis behind this research is that if the activities performed by systems engineers can be plotted in a framework it may be able to bring about a revision of the *a priori* understanding of systems engineering. This means a change in the understanding of its current paradigm (Churchman, 1979) page 105) or weltanschauung (Checkland and Scholes, 1990), and its emergence as a true engineering discipline. Kasser discussed the evolution of a proposed framework for understanding systems engineering (Kasser, 2006). This Chapter places that framework in the context of Kline's definition of a discipline.

21.4 Elements relevant to research in a discipline

Research into a discipline needs the following three items (Checkland and Holwell, 1998):

1. **An Area of Concern** (A), which might be a particular problem in a discipline (area of study), a real-world problem situation, or a system of interest.
2. **A particular linked Framework of Ideas** (F) in which the knowledge about the area of concern is expressed. It includes current theories, bodies of knowledge, heuristics, etc.as documented in the literature as well as tacit knowledge.

3. **The Methodology** (M) in which the framework is embodied that incorporates methods, tools, and techniques in a manner appropriate to the discipline that uses them to investigate the area of concern.

Figure 21-1 (Checkland and Holwell, 1998) page 23) illustrates the relationship between these elements. Given that there is a working community of paid scholars and/or practitioners (Kline, 1995) page 3), these same three elements can also be used to characterize a discipline because they expand Kline's specification and encompass the key aspects of a discipline (Cook, et al., 2003). Consider each of these elements in turn, as they apply to systems engineering.

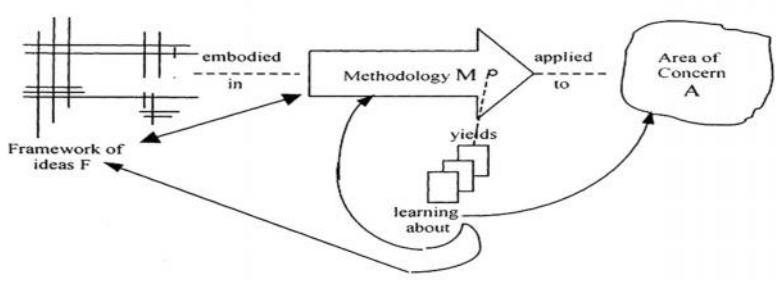


Figure 21-1 Elements relevant to any piece of research (Checkland and Holwell, 1998: p 13)

21.4.1 An area of concern (A)

The Area of Concern (A) should cover what systems engineers do, where they do it, and the overlapping of, and differences in, the roles of systems engineering, systems architecting, and project management. There have been many diverse opinions on these topics over the years, typical examples were discussed in Chapters 2, 12 and 14, hence the difficulty in defining systems engineering. Three more sample opinions are:

- "*Despite the difficulties of finding a universally accepted definition of systems engineering, it is fair to say that the systems engineer is the man who is generally responsible for the over-all planning, design, testing, and production of today's automatic and semi-automatic systems*" (Chapanis, 1960) page 357).
- The principal functions of systems engineering are "*to develop statements of system problems comprehensively, without disastrous oversimplification, precisely without confusing ambiguities, without confusing ends and means, without eliminating the ideal in favour of the merely practical, without confounding the abstract and the concrete, without reference to any particular solutions or methods, to resolve top-level system problems into simpler problems that are solvable by technology: hardware, software, and bioware, to integrate the solutions to the sim-*

pler problems into systems to solve the top-level problem" (Wymore, 1993) page 2).

- "*Systems engineering is a wide-range activity, and it should not be handled in the same form for all kinds of systems*" (Shenhar and Bonen, 1997).

Each opinion seems to represent a viewpoint based on the experience of the writer[90]. In addition, the latest systems engineering standard ISO 15288 provides a list of the organizational processes or activities in which systems engineers are involved as shown in Figure 21-2 extracted from the Standard (Arnold, 2002) page 61). Thus, ISO 15288 could be considered as a framework for systems engineering based on the activities which systems engineers perform.

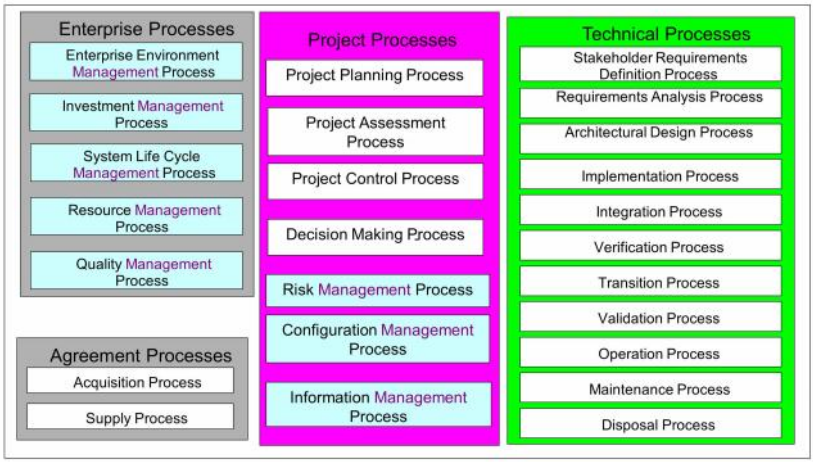


Figure 21-2 ISO 52888 systems engineering processes

Chapter 19 discussed the overlapping of, and differences in, the roles of systems engineering, systems architecting, and project management. The (A) of systems engineering thus needs to span the activities performed by the roles of the systems engineer, operations researcher and project manager.

21.4.2 The framework of ideas (F)

Checkland and Holwell discuss the importance of a "declared-in-advance" epistemological framework (F) when undertaking interpretive research (Checkland and Holwell, 1998) pages 23-25). Thus establishing an (F) is fun-

[90] At least in the case of (Kasser, 1995)

damental to the definition of a research topic or a discipline. As systems engineering focuses on problems (Wymore, 1994)[91] and the common denominator in the definitions of a system listed in Chapter 12 is the statement of a problem, the (F) for systems engineering can be considered as being documented in the literature on critical thinking, systems thinking, problem solving in the activities that take place in the (A)[92].

21.4.3 The methodology (M)

Since the activity known as systems engineering overlaps other organizational activities (Chapter 12), systems engineering may be considered as a meta-methodology incorporating the methodologies, tools and techniques used in the (A) by both systems engineers and practitioners of the other organizational activities[93]. This puts a considerable number of tools into the toolbox of the systems engineer including:

- Total Systems Intervention (TSI) (Flood and Jackson, 1991);
- Soft Systems Methodology (SSM) (Checkland and Holwell, 1998);
- the process-oriented, blended, object-oriented, rapid development, people oriented, and organisational-oriented methodologies discussed in (Avison and Fitzgerald, 2003);
- a whole suite of problem solving tools for requirements elicitation and elucidation (Hari, et al., 2007); these include interviews (Alexander and Stevens, 2002), Joint Applications Development (JAD) (Wood and Silver, 1995), Analytical Hierarchical Process (AHP) (Saaty, 1990), Nominal Group Technique (NGT) (Memory Jogger, 1985) scenario building, user/customer interviews, questionnaires, customer visits, observation, customer value analysis, use cases, contextual inquiry, focus groups, viewpoint modelling (Darke and Shanks, 1997), and Quality Function Deployment (QFD) (Hauser and Clausing, 1988; Clausing and Cohen, 1994);
- the more commonly used hard systems methodologies (Blanchard and Fabrycky, 1981; Buede, 2000) and other treatments of the systems engineering process.

Thus systems engineering has an (A), (M) and (F) and meets (Kline, 1995)'s view of a discipline namely it has "*a specific area of study, a literature, and a working community of paid scholars and/or paid practitioners*" and contains the elements relevant to research in a discipline (Checkland and Holwell, 1998). All it lacks is a framework for understanding it!

---

[91] Wymore was in the problem camp, see section 29.2.3.

[92] This statement led to the research documented in (Kasser, 2013) discussed in the preface to the second edition.

[93] This is the perspective from the meta-discipline camp, see section 29.2.4.

21.5 Requirements for a framework

A framework for understanding systems engineering must provide an improvement on the current paradigm, or there is little point in developing one. Looking around at the state of systems engineering and its problems, the following four requirements for such a framework were defined (Kasser, 2006).

1. The framework shall provide an understanding of why systems engineers can't agree on their roles and activities.
2. The framework shall provide an understanding of the reasons for the overlap between systems engineering and project management.
3. The framework shall provide a way to cope with complexity.
4. The framework shall enable the development of a way of working that lowers the cost of doing work by at least an order of magnitude.

21.6 Rationale for the requirements for the framework

The rationale for these four requirements for the framework is explained below.

21.6.1 The framework shall provide an understanding of why systems engineers can't agree on their roles and activities

Systems engineering had a large number of definitions (Chestnut, 1965) page 8) citing (Churchman, et al., 1957; Goode and Machol, 1959; Morton, 1959; Eckman, 1961; Williams, 1961; Hall, 1962; Gosling, 1962; Bender, 1962; Feigenbaum, 1963; Mesarovic, 1964) and a number of subsequent definitions were presented in Chapter 12. This is a situation characteristic of the forming stages of a discipline and needs to be remedied. Hill and Warfield anticipated George Friedman (Friedman, 2006) writing "*development of a theory of systems engineering that will be broadly accepted is much to be desired*" (Hill and Warfield, 1972). Without such an understanding, systems engineers will continue to discuss rather than develop and apply systems engineering, and not move onwards to the creation of a discipline. Until a framework for the broad range of activities known by the term 'systems engineering' is developed and systems engineers understand their location of their activity within the framework it will be impossible to develop a theory of systems engineering.

21.6.2 The framework shall provide an understanding of the reasons for the overlap between systems engineering and project management

We need this understanding to be able to reengineer organizations in order to remove the overlap. The overlap is expensive due to both the duplication of resources and the modern management paradigm which has separated the decision makers from the people who understand the implications of the decisions. This situation was recognized almost at the dawn of systems engineering by Goode and Machol who wrote "*The most difficult obstacle that may be encountered by an [systems] engineer is not the problem but a management which is unsympathetic or lacking in understanding*" (Goode and Machol, 1959) page 513). The optimal management method is said to be "*Management by Walking Around*" (MBWA) (Peters and Austin, 1985). Yet Deming wrote *"MBWA is hardly ever effective. The reason is that someone in management, walking around, has little idea about what questions to ask, and usually does not pause long enough at any spot to get the right answer"* (Deming, 1986) page 22). And the situation continues into the 21st century as satirized by Scott Adams in his Dilbert cartoons (Adams, 2006). Think of the cost of the waste and the work expended to implement and then correct the results of poor decisions. Once there is an understanding of the reasons for and the nature of the overlaps, we stand a chance of removing the overlap.

21.6.3 The framework shall provide a way to cope with complexity

Systems engineering has not delivered on its promise to meet the challenge of complexity as documented by Chestnut who wrote "*Characteristic of our times are the concepts of complexity, growth and change*" (Chestnut, 1965) page 1) and "*in a society which is producing more people, more materials, more things, and more information than ever before, systems engineering is indispensable in meeting the challenge of complexity*" (Chestnut, 1965) page vii). There is a growing dichotomy in the literature on the subject of complex systems. On one hand there is literature on the need to develop new tools and techniques to manage them, e.g. (Cook, 2000; Bar-Yam, 2003). On the other hand, there is literature on techniques such as aggregation which mask the underlying complexity to ensure that only the pertinent details for the particular situation to deal with the issues are considered e.g. (Hitchins, 1998; Maier and Rechtin, 2000; Hitchins, 1992). Perhaps the dichotomy is due to the observation that "*the classification of a system as complex or simple will depend upon the observer of the system and upon the purpose he has for considering the system*" (Jackson and Keys, 1984). For the framework to be useful, it must provide a way to cope with complexity.

21.6.4 The Framework shall enable the development of a way of working that lowers the cost of doing work by at least an order of magnitude

This is a grand target to aim at – more of a goal than a practical requirement. Systems engineering overlaps project management and other organizational activities (Johnson, 1997; Roe, 1995), so it can adopt some of their tools and techniques. There should be no reason why the process architecting function discussed in Chapter 19 could not improve processes, products and organizations to the point where the cost of developing projects is lowered by an order of magnitude. If the Framework only allows the achievement of 50% of this target, it will still be a significant improvement.

Table 21-1 Methodologies in the problem context

	Unitary	Pluralist
Simple	Operations research Systems analysis Systems engineering Systems dynamics	Social systems design Strategic assumption surfacing and testing
Complex	Viable system diagnosis General system theory Socio-technical systems thinking Contingency theory	Interactive planning Soft systems methodology

21.7 Candidate Frameworks

The first candidate was based on the work of Flood and who developed a systemic meta-methodology called TSI that guides practitioners through a systemic process of choosing a methodology based on the problem situation Jackson (Flood and Jackson, 1991). TSI is broadly summarized in Table 21-1 (Flood and Jackson, 1991) page 42). However, application of the Table is not as simple as it appears since "*the classification of a system as complex or simple will depend upon the observer of the system and upon the purpose he has for considering the system*" (Jackson and Keys, 1984)[94]. This prompted

[94] And the confusion between complex and complicated discussed in Chapter 7 of (Kasser, 2013).

the attempt to identify an alternative framework.

Building on prior research, the next set of candidates for a framework for understanding systems engineering were the five identified in Chapter 12 namely:

1. Allison and Cook,
2. Hitchins' Five-layer Model,
3. Sage's Three Overlapping facets Model,
4. Badaway's Master of Technology, and
5. Kasser's PPPT enterprise framework.

21.8 Evaluating the frameworks against the requirements

Chapter 12 showed that Hitchins framework extended over time into the two-dimensional HKMF contained the other candidates. The Chapter also stated that by combining the PPPT and Hitchins models into a multi-dimensional framework, it should be possible to define a framework for discussing and understanding systems engineering.

21.9 The Hitchins-Kasser-Massie Framework

This section introduces a three dimensional framework for discussing and understanding systems engineering. The vertical and horizontal dimensions of the HKMF were introduced in Chapter 12.

21.9.1 The vertical dimension

The vertical dimension is Hitchins five layers of systems engineering (Section 12.1.2).

21.9.2 The horizontal dimension

The horizontal dimension of the framework is organized as sequential phases in providing a whole complete solution to a problem as an overall, end-to-end process which consists of conceiving a whole solution to solve a problem and making that whole "come to life" for the development of a single system in isolation. The phases have been stated in various ways in various Standards, conference papers and books, but in the HKMF they are defined in generic terms as:

A. Identifying the need.
B. Requirements analysis.
C. Design of the system.
D. Construction of the system.
E. Testing of the system components.
F. Integration and testing of the system.

G. Operations, maintenance and upgrading the system.
H. Disposal of the system.

Consider each of them[95].

A. **Identifying the need**. This is the phase where the bulk of the set of activities known as systems engineering is performed. Yet in the Type II[96] systems engineering educational paradigm it tends to be glossed over. Phase A contains the early stage systems engineering activities addressing the problem and determining the conceptual solution and is based on Hall, Gelbwaks and Hitchins (Hall, 1962; Gelbwaks, 1967; Hitchins, 1992) and the summary in Brill (Brill, 1998) and contains the first 'systems engineering' process addressing the conceptual solution. Phase A comprises the following sub-phases:
   1. This sub-phase contains the set of activities that explore/scope the problem, leading directly to Phase A.2. The activities performed in this phase produce a definitive statement of the problem-in-context.
   2. This sub-phase contains the set of activities that conceive the whole solution system (which 'emerges' from/"complements" the problem) and produces the concept of operations (CONOPS) that describes how the solution system will operate in its future environment.
   3. This sub-phase contains the set of activities that design the whole solution system, identify the environment, other interacting systems, the subsystems, parts, interactions, functional architecture, physical architecture, etc., etc., - but still all of the whole.

B. Requirements analysis. This phase is the first phase of the second 'systems engineering' process addressing the physical solution and its implementation and contains the set of activities that specify the solution system as a full set of specifications for the whole and for the parts and their infrastructure, including the environment/weltanschauung or paradigm that justifies them. If the specifications are in the form of text mode requirements, the output of this phase tends to be at the 'A' specification level (MIL-STD-490A, 1985). Unfortunately, many systems engineers have been educated to consider this phase as the first phase of a single systems engi-

[95] This section was originally published in what is now Chapter 23 after the first edition of this book had gone to press and has been relocated here in the second edition.
[96] Or B paradigm discussed in Chapter 28.

neering process[97]. For example, (1) requirements are one of the inputs to the 'systems engineering process' (Martin, 1997) page 95), (Eisner, 1997) page 9), (Wasson, 2006) page 60) and (DOD 5000.2-R, 2002), pages 83-84); and (2) in one postgraduate class at UMUC the instructor stated that systems engineering began for him when he received a requirements specification (Todaro, 1988). While DOD does call out the 'analysis of possible alternatives' subset of activities in Phase A2 of the HKMF 5000 (DOD 5000.2-R, 2002) pages 73-74), those activities are called out as part of the separate seemingly independent CAIV process which (1) is a way of complicating just a part of the concept of designing budget tolerant systems using the Cataract approach (Chapter 13) and (2) takes place **before** the DOD 5000.2-R 'systems engineering process' begins.

C. Design. This phase contains the set of activities that creates a more detailed design of the whole solution system through a combination of people, doctrine, parts, subsystems, interactions, etc., including configuration, architecture and implementation criteria. The output of this phase tends to be at the 'B' specification level (MIL-STD-490A, 1985).

D. **Construction.** This phase contains the set of activities that create the individual parts, subsystems, interactions, etc. *in isolation.* Consequently the set of activities are mainly engineering, training, etc., not systems engineering. This situation is indicated in Figure 23-1 by the down slope in the line showing the amount of systems engineering at this phase.

E. Unit Testing. This phase contains the set of activities that validate the performance of the individual parts, subsystems, interactions, etc. *in isolation* against their requirements. Consequently the set of activities are mainly engineering, not systems engineering.

F. **Integration and testing of the system.** This phase contains the set of activities that (1) combines the parts, subsystems, interactions, etc., to constitute the solution system, and (2) establishes, under test conditions, the performance of the whole solution system, with optimum effectiveness, in its operational context.

G. **Operations, maintenance and upgrading of the system.** This phase contains the set of systems engineering and non-systems engineering activities that actively provide a solution to the problem for which the whole system was created. This phase includes operating the system, support to maintain operations, improvements to the whole to enhance effectiveness, and to accommodate changes in

[97] See Chapter 28

the nature of the problem over time. These changes iterate phases A to F (call them Ga ... Gf), ideally without rendering the operating solution system materially inoperative for an unacceptable period of time.

H. **Disposal of the system.** This phase contains the set of activities that dispose of the system. This phase is rendered necessary where either where the problem no longer exists, or the solution system is no longer capable of solving the problem effectively or economically. If the disposal method has not been predetermined, this phase may also iterate phases A to F (call them Ha ... Hf).

The resulting two-dimensional framework is shown in Figure 21-3[98]. The HKMF's vertical and horizontal dimensions provide a map for the location of the activities performed by systems engineers.

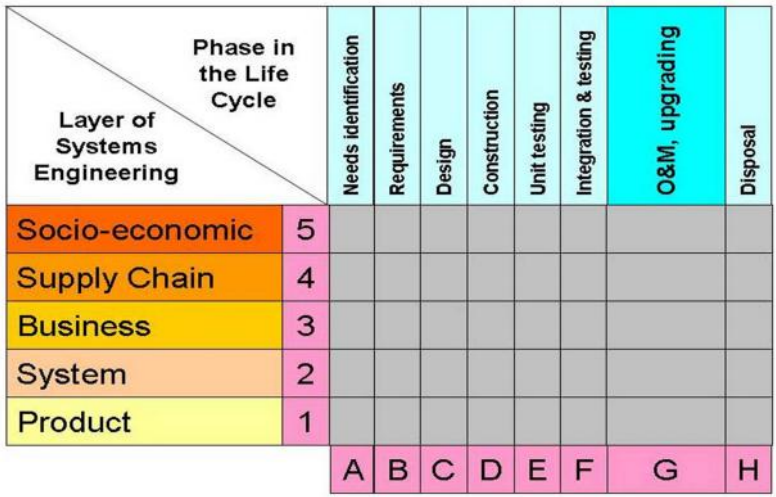


Figure 21-3 The HKMF for understanding systems engineering

This Chapter now goes beyond Chapter 12 and discusses the development of the candidate for the third dimension. The third dimension of the HKMF is the difficult one since there are many ways to classify the types of problems posed in each area of the network. One immediately obvious approach is by the domain (aerospace, military, commercial, etc.); however applying systems thinking and incorporating lessons learned in other domains, it was felt that:

[98] Credit is due to Ms. Xuan-Linh Tran at the Systems Engineering and Evaluation Centre (SEEC) in the Univsersity of South Australia (UniSA) for drawing the framework in this format.

- this situation was analogous to the development of theories of motivation in Psychology, and
- if the analogy holds true then applying lessons learned from Psychology to systems engineering, should provide a workable framework.

At one point of time in the development of theories of motivation, Henry A. Murray identified separate kinds of behaviour and developed an exhaustive list of psychogenic or social needs (Murray, 1938). However, the list is so long that there is almost a separate need for each kind of behaviour that people demonstrate (Hall and Lindzey, 1957). While this list has been very influential in the field of psychology, it has not been applied directly to the study of motivation in organizations. This is probably because the length of the list makes it impractical to use. On the other hand, Maslow's hierarchical classification of needs (Maslow, 1954; 1968; 1970) has been by far the most widely used classification system in the study of motivation in organizations. Maslow differs from Murray in two important ways; his list is:

- **Arranged in a hierarchy** -commonly drawn as a pyramid, and contains a set of hypotheses about the satisfaction of these needs.
- **Short** -- Only five categories.

Clayton P. Alderfer subsequently proposed modifying Maslow's theory by reducing the number of categories to three (Alderfer, 1972). Murray's and early theories defined needs or instincts; Maslow's shows interdependencies and relationships between those needs and Alderfer proposed further reductions in the number of categories. Applying this situation to systems engineering, it was felt that using system domains as the third dimension would be analogous to using Murray's list of needs and a Maslow/Alderfer more generic-type classification was needed. Consider Maslow as having identified common categories and then grouped Murray's needs into those categories as well as adding the interdependencies and relationships between those needs. The different domains could be likened to Murray's lists, and since in any domain of systems engineering systems engineers deal with problems (Wymore, 1994) a short classification of problems could be likened to Maslow's categories. Thus, while the levels of complexity and types of system can also be expected to be influential in systems engineering, the first attempt to formulate a framework for systems engineering in this research based the third dimension of the framework on problem solving (risk mitigation) (Chapter 12). One context of categories for risk mitigation found in the literature presented a taxonomy in which systems were classified according to three levels of system scope and four levels of technological uncertainty (risk) (Shenhar and Bonen, 1997). Their three levels of system scope correspond roughly to the three lower layers of the Hitchins five layer model (Section 12.1.2) and their four levels of technological uncertainty (risk) are:

- **Type a**[99] — Low-Technology Projects which rely on existing and well-established technologies to which all industry players have equal access. The system requirements of Low-Tech Projects are usually set by the customer prior to signing the contract and before the formal initiation of the project execution phase.
- **Type b** — Medium-Technology Projects which rest mainly on existing technologies; however, such systems incorporate a new technology or a new feature of limited scale. Their requirements are mainly set in advance; however, some changes may be introduced during the product development phase. This process often involves a joint effort of the contractor and customer. It may also require the involvement of potential customers in the process.
- **Type c** — High-Technology Projects which are defined as projects in which most of the technologies employed are new, but existent — having been developed prior to the project's initiation. System requirements are derived interactively with a strong involvement by customers or potential users, and many changes are introduced during the development phase.
- **Type d** — Super-High-Technology Projects which are based primarily on new, not entirely existent, technologies. Some of these technologies are emerging; others are even unknown at the time of the project's initiation. System requirements are hard to determine; they undergo enormous changes and involve extensive interaction with the customer.

The differences between the four types of projects are summarized in Table 21-2.

As the development progresses through the systems development lifecycle the work takes place in different areas of the HKMF. The nature of the problems faced by systems engineers in each area of the framework will be different because the problems will depend on the level of technological uncertainty of the specific system (Shenhar and Bonen, 1997). For example, a systems engineer could be working in Area '2Ba' if it is a low technical risk system or in Area '2Bd' if it is a Super-High-Technology Project. Shenhar and Bonen stated that [the role of] systems engineering was a wide-ranging activity, and should not be performed in the same manner for all kinds of systems. Shenhar and Bonen also claim that adopting the wrong system and management style may cause major difficulties during the process of system creation. Namely what works in Area '2Ba' may not work in Area '2Bd'.

[99] Shenhar and Bonen used upper case letters to designate the types. This text uses lower case to differentiate them from the upper case designator of the horizontal dimension.

Table 21-2 Shenhar and Bonen's project classification by technology uncertainty

	Type a	Type b	Type c	Type d
	Low - Tech	Medium - Tech	High - Tech	Super – High - Tech
Technology	All exist	Integrates some new with mostly existing	Integrates mostly new with some existing	Key technologies do not exist at project's initiation
Development	None	Some	Considerable	Extensive
Testing	None	Some	Considerable	Extensive
Prototyping	None	Some	Considerable	Extensive
Requirements	Known prior to project start	Joint development effort between customer and contractor	Strong involvement of contractor	Extensive contractor involvement many changes and iterations
Design cycles	1	1 or 2	At least 2	2 to 4
Design freeze	Prior to project start	1st Quarter	1st or 2nd Quarter	2nd or 3rd Quarter
Changes	None	Some	Many	Continuous
Management and systems engineering style	Firm and formal	Moderately firm	Moderately flexible	Highly flexible

21.10 Meeting the requirements for the framework

For the HKMF to offer a serious competitive advantage over the current paradigm it must meet all of the four requirements for the framework stated above. Consider ways in which the HKMF meets or can meet the requirements.

21.10.1 The framework shall provide an understanding of why systems engineers can't agree on their roles and activities

Sheard discussed two perspectives on twelve roles of systems engineers some of which were life-cycle roles, some program management, and some which belonged in both (Sheard, 1996). The framework places that discus-

sion in context of the areas in which the activities are performed, because systems engineers work in different layers and in different phases of each layer. Cook and Kasser used an early representation of the HKMF combined with the V view shown in Figure 21-4 in the classroom to position the areas in which systems engineers work as summarized below (Cook, 2003):

- Traditional systems engineering covers Layer 2 completely as shown in Figure 21-5.
- Contemporary test and evaluation is shown in Figure 21-7. The "V" representation can be seen in the figure.
- Military platforms lie mostly in Layer 2 with some activities in Layers 3 and 1 as shown in Figure 21-6.

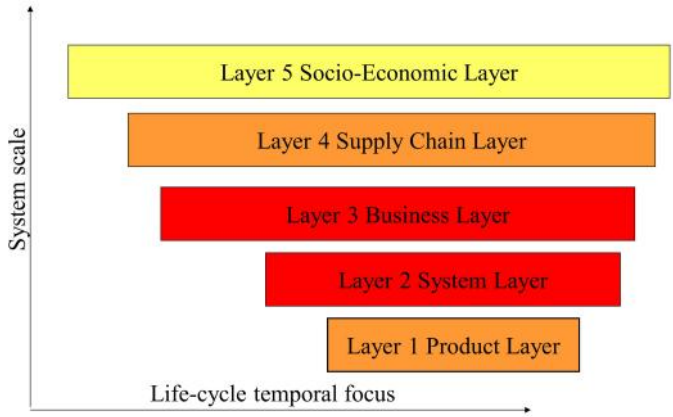


Figure 21-4 Cook's classroom version of the HKMF

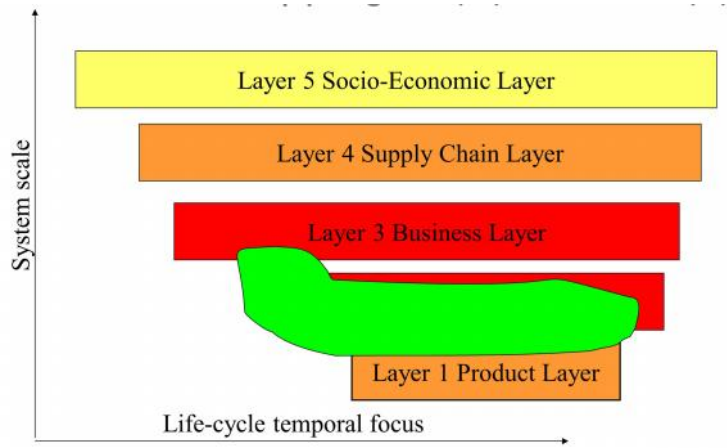


Figure 21-5 Traditional systems engineering

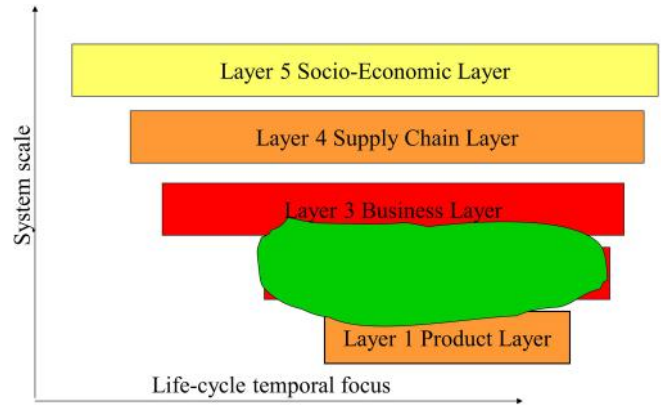


Figure 21-6 Military platforms

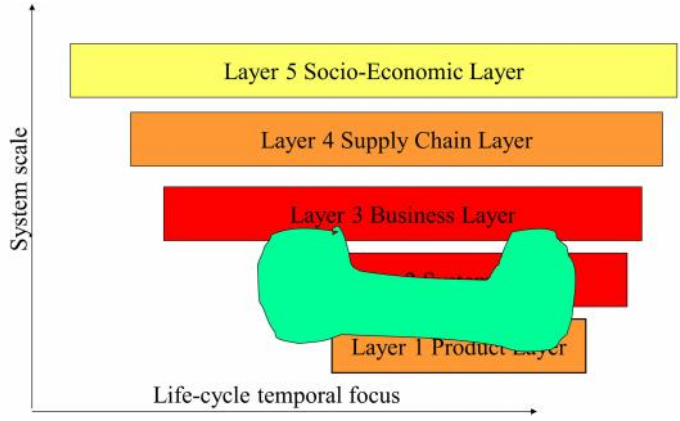


Figure 21-7 Contemporary test and evaluation

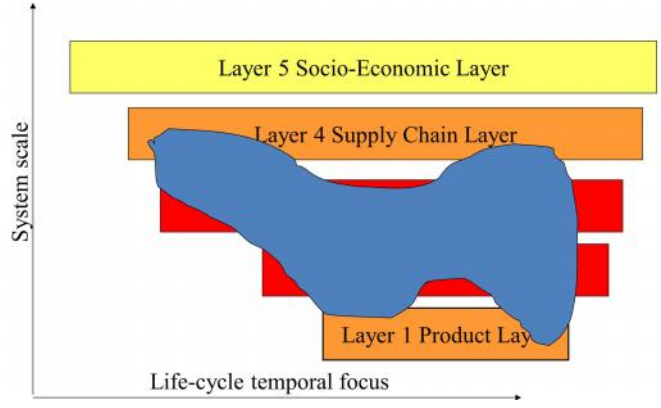


Figure 21-8 Information systems

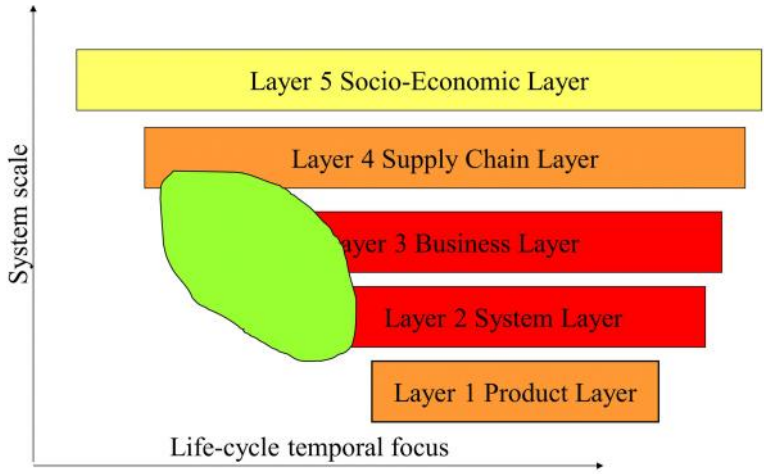


Figure 21-9 Capability development

- Information systems overlap several layers as shown in Figure 21-8. They comprise traditional systems integrated out of products interacting with the business and supply chain layers.
- Capability Development lies as shown in Figure 21-9. These activities roughly correspond to the investment management and resource management processes shown in Figure 21-2 (Arnold, 2002). The positioning of Capability Development in the figure indicates that this activity is focussed in the front of the business-layer lifecycle. Capability Development also interacts with the supply chain level because there is a need to ensure enduring support to future Defence capabilities. Lastly, it interfaces to Layer 2 through the acquisition projects it generates[100].

When activities which were plotted in the framework are overlapped, the result is shown in Figure 21-10. It shows that systems engineers working in the different parts of the framework do different tasks. Figure 21-10 does not show, but other evidence indicates that the systems engineers working in the different areas of the framework use the same words but with different meanings. For example the word "capability" has different meanings in areas '3A' and '2C'. Consequently, no wonder they can't agree on what systems engineering is and on what systems engineers do.

[100] According to (Cook, 2003), such a representation is, of course, overly simplistic because aspects of the capability development processes also occur further down the life-cycle, thus a more accurate representation would be an overlay whose colour saturation represents the degree of effort applied at each point in the two-dimensional space.

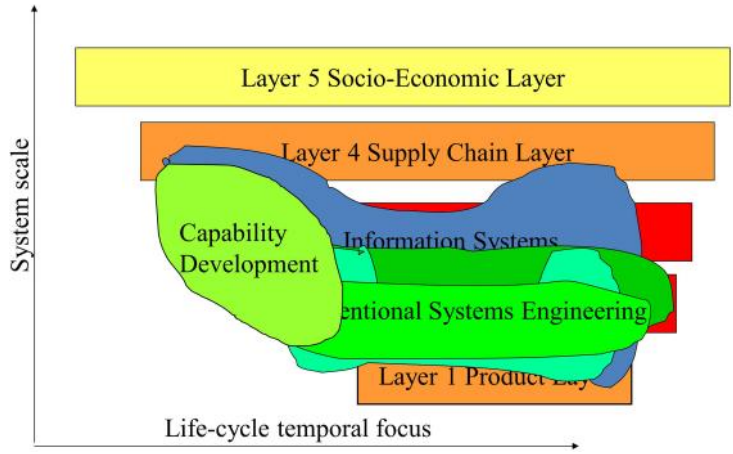


Figure 21-10 Overlay of areas

As if this isn't enough, consider the role of the systems engineer in Layer 2 which was examined (Chapter 18). Systems engineers working in that layer do different things in different organizations due to the overlap of roles and activities, and there is no reason to think that systems engineers working in other layers also do different things in those layers than do systems engineers working in different organizations. Again it should not come as a surprise that systems engineers working in the same layer but in different organizations can't agree on what systems engineers do and what systems engineering is without this external perspective provided by the framework. Thus it can be said that the HKMF meets the requirement to provide an understanding of why systems engineers can't agree on roles and activities.

21.10.2 The framework shall provide an understanding of the reasons for the overlap between systems engineering and project management.

The overlap was discussed above. Research into the reason for the overlapping of the disciplines turned up information as to how the overlap originated in the form of the following statement "*Driven by cold war pressures to develop new military systems rapidly, operations research, systems engineering, and project management resulted from a growing recognition by scientists, engineers and managers that technological systems had grown too complex for traditional methods of management and development*" (Johnson, 1997)

Thus systems engineering, project management and operations research can be seen as three solutions to the problems of the Cold War by three different communities of practice (Johnson, 1997) that have continued to evolve and overlap. In specific organisations, practitioners of one of the disciplines would perform activities that were not being performed in that

organisation, but were being performed by a practitioner of a different discipline in a different organisation (Chapter 19). As a result,

- Today's organisational paradigm contains three overlapping evolving disciplines (project management, systems engineering and operations research) attempting to solve the same problems from three different perspectives (Johnson, 1997).
- Each discipline is using its own tools and techniques and adopting others as and when needed.
- Each discipline has instances of poor implementation leaving a vacuum which another discipline fills. This has created a large degree of overlap of activities. These boundaries are artificial and often detrimental (Friedman, 2006).
- The evolution and overlap is continuing. Note the eight boxes containing the word "management" in Figure 21-2.
- The overlap between operations research and systems engineering was noted as early as 1954 when Johnson wrote "*Operations research is concerned with the heart of this control problem – how to make sure that the whole systems works with maximum effectiveness and least cost*" (Johnson, 1954) page xi) a goal that many modern systems engineers would apply to systems engineering. Goode and Machol wrote that the steps of the operations research and systems engineering processes have much in common however there is a fundamental difference in approach namely "*the operations analyst is engineer is primarily interested in making equipment changes*". A lasting difference was noted by Roy as "*Operations research is more likely to be concerned with systems in being than with operations in prospect*" (Roy, 1960) page 22).

By mapping the activities performed by the three disciplines involved in acquiring and maintaining systems into a two-dimensional map, the framework has confirmed that organisational activities do overlap and has initiated research into the history of the disciplines which has provided further information as to how the overlaps began and evolved (Johnson, 1997). The framework can thus be said to have met this requirement.

21.10.3 The framework shall provide a way to cope with complexity.

Systems engineering has been the promised approach to solving the problems of complexity for at least 50 years. For example, Chestnut wrote in 1965 "*In a society which is producing more people, more materials, more things, and more information than ever before, systems engineering is indispensable in meeting the challenge of complexity*" (Chestnut, 1965) page vii). About twenty years later, Wymore wrote "*Systems engineering is the professional, intellectual and academic discipline the primary concerns of which are the analysis and design of large-scale, complex, man/machine systems*"

(Wymore, 1976). Yet instead of solving problems, systems engineering seems to be making things more and more complex.

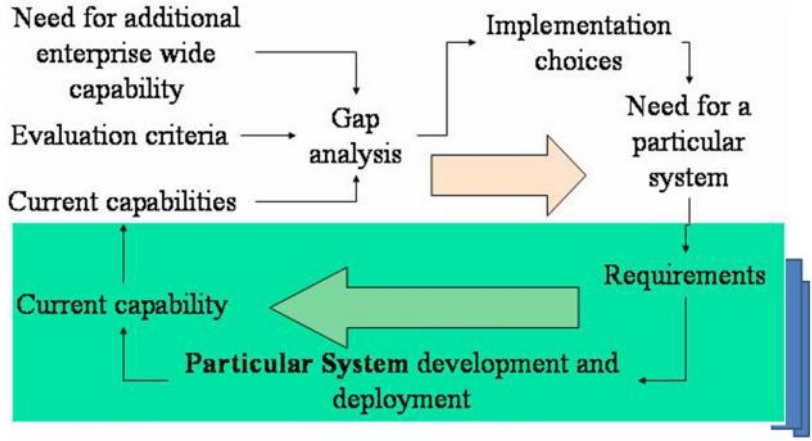


Figure 21-11 The process for the engineering of complex systems

Consider the systems acquisition process from the perspective of the HKMF. The top half of Figure 21-11 represents a reference model for the activities performed in the acquisition process in the top three layers of the HKMF. The cycle begins when a high level need for additional enterprise wide Defence Capability is identified. It is compared with the Capability that exists or is in the process of being acquired and due to be phased into service, and a gap analysis is made to identify missing Capability. Once identified, the missing Capability is slated to be acquired. To guide the acquisition process, evaluation criteria are also developed to influence the acquisition decision. Ideally the gap analysis generally identifies a number of implementation choices or possible solutions to the problem posed by the missing Capability. These choices include not only the procurement of new materiel but changes to doctrine and/or operations and reuse of spares located somewhere else in the enterprise. Recent changes to the acquisition process perform the gap analysis to not only identify current Capability gaps, but project into the future to identify the most probable gaps, five, ten and even twenty years into the future producing a lot of PowerPoint engineering products and science fiction. Thus what has been termed as Enterprise Systems Engineering and the Engineering of Systems applies across many systems. It:

- Converts capability needs to requirements.
- Identifies problems.
- Mostly uses soft systems methodologies.

- Tends to be political rather than technical.
- Employs an enterprise wide architecture framework.
- Comprises the top three layers of the framework.

Glossing over how the decision of which specific choice to implement, a decision is made and a specific system is defined and enters the acquisition process, namely:

- Contract(s) are awarded for development of the Capability the system is to provide;
- Changes are managed throughout the SDLC of the Capability being acquired[101].

The lower half of Figure 21-11 represents one instance, namely the particular system of many such instances of Capability being acquired and upgraded asynchronously (Chapter 14) in such a manner as to ensure the entire available Capability meets the need(s) all the time. Thus what has been termed as traditional systems engineering:

- Applies to particular or single systems.
- Converts requirements to Capability.
- Implements solutions.
- Mostly uses hard systems methodologies.
- Tends to be technical rather than political.
- Shows how a particular system fits into the enterprise-wide architecture framework.
- Takes place in the lowest two layers of the HKMF.

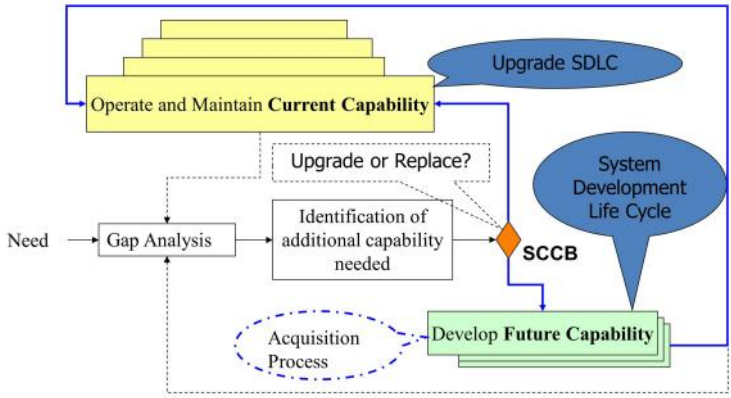


Figure 21-12 A service and support system

[101] This is a description of a reference model; the real world is somewhat different.

If the outward looking perspective shown in Figure 21-11 is replaced by an external perspective, the situation can be represented in the form of the reference model shown in Figure 21-12. When drawn in this way several things become apparent, namely:

- Figure 21-12 is a representation of a complex system, yet is understandable in accordance with the Simplicity paradigm (Chapter 18).
- The purpose of drawing the system boundaries for the reference model shown in Figure 21-12 can be stated as a high level representation of a system that provides Capability as and when needed. The algorithm of the system is when a need for Capability arises, a gap analysis is made between the Capabilities that are available and those planned to go into service (within their individual SDLC) and a decision is made to fill the gap by upgrading existing Capability, replacing Capability or providing new Capability.
- The box marked "gap analysis" could also be labelled "strategic planning", the box marked "identification of additional capability needed" could be labelled "capability development" and the box marked "future capability" could be labelled as "produce future capability". In this context, Figure 21-12 is a functional representation of a system.
- System architecting and systems engineering activities can be seen taking place in three parts of Figure 21-12 (Upgrade, Upgrade or Replace, and the SDLC).
- The methodologies used in various parts (subsystems) of Figure 21-12 are different (Flood and Jackson, 1991; Shenhar and Bonen, 1997; Martin, 1994). Strategic planning, capability development and requirements elicitation in the upper layers of the HKMF tend to be performed in a pluralist context, while once the requirements for a system are known, the context generally shifts to unitary and traditional systems engineering in Layer 2 of the HKMF takes place. Once the context of the problem is known the appropriate methodology can be determined by the process architect (Chapter 19).

After thousands of years of performing physical decompositions of systems we have a good understanding of the process and can do it very well to simplify physical systems. We need to develop ways of performing non-physical decompositions of systems in a similar manner to simplify problems instead of introducing additional complexity. The framework has helped identify the problem and methodologies, but much more research needs to be done to convert complexity to simplicity.

21.10.4 The framework shall enable the development of a way of working that lowers the cost of doing work by at least an order of magnitude

Meeting this requirement will take a real reengineering effort (Hammer and Champy, 1993). It will need a fundamental change in the structure of the organisation and the partition of work. Understanding the reasons for the overlapping of roles in the organization allows the roles to be redefined to minimize the overlaps in the future. Redrawing the boundaries can be implemented as part of an object-oriented approach to building organizations.

As a systems engineer, it is tempting to state that systems engineering is the methodology to use to reduce the cost of doing work. However, as Maslow wrote, "*I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail*" (Maslow, 1966) pages 15 and 16). Thus applying systems thinking tells us to use the various methodologies, tools and techniques used by any of the overlapping activities of systems engineering, project management, operations research, TQM, and others in the area of activity covered by the HKMF. The challenge is to build a new paradigm for doing work at a lower cost (the system) by mixing components from:

- Systems engineering (roles and activities) (Beer, Hall, Jackson, Checkland etc.);
- Management (Taylor, Ford, Drucker, Peters, Hammer and Champy etc.);
- Quality (Deming, Juran and Crosby etc.);
- Other appropriate sources;
- Some original twists.

Early activities using an object-oriented approach to redraw boundaries and include prevention into the work process was described as an Anticipatory Testing approach (Kasser, 1995) and achieved promising results as shown by the following examples of the difference between the current way of working in general, and the Anticipatory Testing approach. In the period September – December 1995, the Anticipatory Testing Corporation took part in three proposal efforts, these were the:

- **PK proposal** - A 75 page technical proposal (+ other volumes) which was written both in Maryland and in Florida. The Anticipatory Testing Corporation led the proposal preparation effort which lost by being $10,000 more expensive than the winning offer.
- **NOVA proposal** - A 100 page technical proposal (+ other volumes) which was written in Maryland. The Anticipatory Testing Corporation was a small sub-contractor, had no say in the proposal preparation effort.
- **NSF proposal** - A 147 page technical proposal (+ other volumes) which was written both in Maryland and Virginia in December 1995. The Anticipatory Testing Corporation led the proposal preparation effort.

The estimated total costs for each proposal are the costs for labour and materials assuming everyone who worked on the proposal were paid for all the hours they put in to preparing the proposal. A comparison of the PK and NOVA proposals is shown as part of Table 21-3. The 10 to 1 difference in cost is mostly the result of the Anticipatory Testing management approach rather than the size or type of the proposals. The NSF proposal provided another data point which correlates to the PK proposal.

Table 21-3 Comparison of September – December 1995 proposals

Factor	NOVA	PK	NSF
Technical pages	100	75	147
Companies on team	4	2	3
Location	MD/DC	MD/FL	MD/VA
Management approach	Conventional	Anticipatory testing	Anticipatory testing
Estimated total equivalent costs ($US)	$100,000	$10,000	$20,000

The same Anticipatory Testing approach was later used in 2003 in the SEEC at the UniSA to write the main 32 pages of a proposal[102] to provide the Australian Defence Materiel Organisation (DMO) with a US style coursework Master of Project Management degree with flexible delivery options. SEEC had no prior contracts with the DMO, so it was a cold proposal. The proposal effort began half way into the six-week tendering period and was mostly written by one person. It was evaluated by the DMO customer as providing the best value, coming ahead of eight competing Australian universities, and the contract was worth more than AUD $1,500,000 over three years.

Redrawing boundaries has introduced the concept of the process architect (Chapter 19) and has also identified defects in the systems engineering process, which if fixed can substantially contribute to lowering the cost of doing work. These defects, and how to fix them, were discussed in Chapter 20. Thus while it cannot be claimed that the framework has met this requirement, it has certainly enabled the understanding of ways in which to do so.

[102] The remaining pages were enclosures containing mostly pre-existing materials.

21.11 Other insight from the framework

The following insight into systems engineering has already been obtained from the HKMF

- A roadmap for a more complete set of system requirements.
- The place of operations research in the framework.
- The similarity between new product development and systems engineering.
- An explanation of the iterative nature of systems engineering.

Consider each of them in turn.

21.11.1 A roadmap for a more complete set of system requirements

The traditional requirements definition process focuses on the functional and performance of the system in its operations and maintenance phase. Treat the HKMF as a landscape underlying a road map, where the activities performed in the SDLC for a specific system begin in columns A or B[103], and end in columns G or H. This gives an explicit assumption that the system will pass through other areas during its SDLC. During the requirements definition phase (Area 2B) determine if any of these areas lay requirements on the system (e.g. supply chain requirements such as installation, storage, and transport), and if so include them in the system requirements.

21.11.2 The place of operations research in the framework

As stated above, "*Operations research is more likely to be concerned with systems in being than with operations in prospect*" (Roy, 1960) Page 22). Thus, operations research seems to focus on the areas in column G of the HKMF.

21.11.3 The similarity between new product development and systems engineering

Priest and Sánchez provide a description of the product development process as follows (Priest and Sánchez, 2001):

- Requirements definition.
- Conceptual design.
- Detailed design.
- Test and evaluation.
- Manufacturing.
- Logistics, supply chain, and environment.

[103] See Chapter 28.

The description of each step in the product development process maps into the HKMF Layer 2 SDLC yet the words systems engineering do not appear in the book. For example, the U-diagram (Priest and Sánchez, 2001) page 64) is equivalent to systems engineering's V-diagram. This is hardly surprising as the product development process takes place in HKMF Layer 1 while traditional systems engineering takes place in Layer 2 of the HKMF.

21.11.4 An explanation of the iterative nature of systems engineering

Students have had trouble grasping the concept that systems engineering is iterative. The Egg diagram (ANSI/EIA-632, 1999) shown in Figure 21-13 shows iterative activities but does not clearly show how the emphasis changes over the lifecycle. The Functions Requirements Answers and Test (FRAT) cycle (Mar, 1994) on the other hand provides a clearer perspective in the context of the HKMF at least for students at the SEEC in the UniSA (Kasser, et al., 2007)[104]. For example, the answer to the question of "when do we do functional analysis?" is "several times". We use it in the 'F' step of the FRAT cycle in each area of the HKMF.

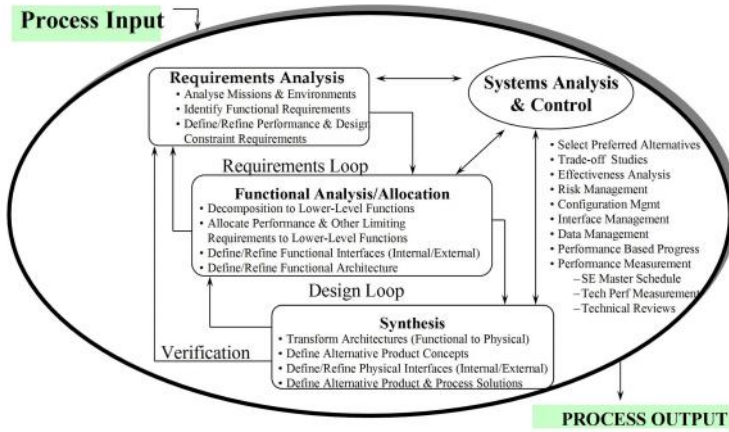


Figure 21-13 ANSI/EIA-632 egg diagram

21.12 Further Research

The HKMF has helped to develop an understanding of the overlap of organisational activities in the workplace and the problems faced in each area of the framework. From there it should be possible to use tools and techniques from all the disciplines in an interdependent manner to design effective or-

[104] See Chapter 22 for an example.

ganisations and reduce the cost of doing work. Early research into redrawing work boundaries has already produced some significant reductions in the cost of doing work. Examples of further research using this framework planned and in process at the present time are:

- Mapping tools used for systems engineering into the areas of the framework. As discussed above, Johnson showed how and why the activity known as systems engineering overlaps other organizational activities (Johnson, 1997). Thus, a multi-methodology for systems engineering can be the methodologies, tools and techniques used in systems engineering as well as any of the overlapping activities namely the (A). This puts the considerable number of tools discussed above into the toolbox of the systems engineer. However, just reusing those tools from the other overlapping areas of activity is as fraught as the reuse of software without investigating the context from where the proposed reusable module was taken from and its suitability for use in the new context. The danger of such software reuse was demonstrated in the failure of the maiden flight of the Arianne 5 launcher on 4 June 1996. Once the context of the problem is known the appropriate methodology to be investigated for suitability needs to be determined by the process architect. This research maps systems engineering tools into the HKMF based on the nature of the problems for which they are used to help provide solutions. This will be followed by mapping tools used in the other overlapping disciplines into the framework and developing requirements for an integrated digital environment or network centric commercial management information system (Tran and Kasser, 2007).
- Plotting the papers published at the annual INCOSE Symposia and regional conferences in the HKMF. In some preliminary research, the topics covered by papers in the Annual International Symposia of the INCOSE published in 1994, 1997, 2000, 2003 and 2006 were plotted into the appropriate areas of the HKMF (Tran and Kasser, 2007). When a paper covered more than one area it was counted in all the areas it covered. Papers that covered topics such as education or theory of systems engineering were excluded. The preliminary findings shown in Figure 21-14 indicate that the papers mostly cover the traditional systems engineering Layer 2 (as shown in Figure 21-5) with a growing interest in Layer 3. This corresponds to growing awareness of the last few years of the need to consider integrating what used to be considered as separate systems and are now being called Systems of Systems.
- Identifying common types of project lifecycles other than the classic Defence SDLC by plotting the path taken by various projects in the framework. One wonders if the paths would be linear or more like the route taken by a player in the 'snakes and ladders' board game.
- Plotting the ANSI/EIE 632, IEEE 1220, ISO 15288, MIL STD 499C standards and the CMMI into the HKMF.

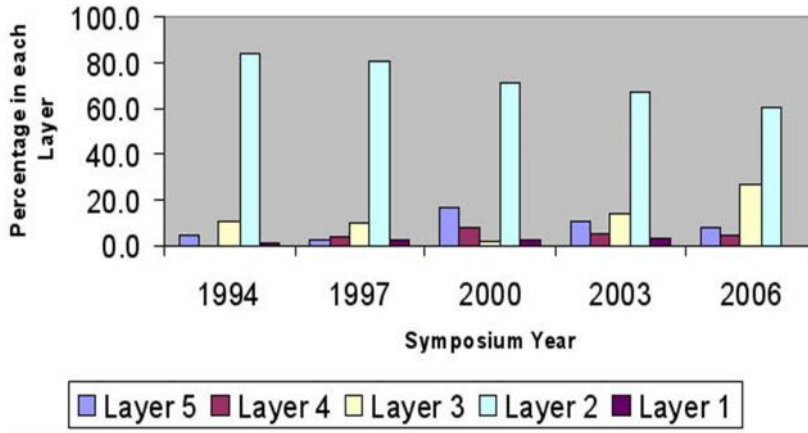


Figure 21-14 Focus of INCOSE symposia papers 1994-2006

- Determination if the Shenhar and Bonen (Shenhar and Bonen, 1997) taxonomy applies in the upper levels of the framework, or if a different type of risk should be considered. The lower two levels apply to the technological implementation phase and technical risk is a logical candidate for this dimension. However, for the upper levels the dimension might be based on contextual risk rather than technical risk.

21.13 Summary

This Chapter applied holistic thinking to systems engineering. This Chapter focused on what systems engineers do. It documented past research and success, and the application of an object-oriented approach in a creative and innovative manner. It listed four requirements for a framework for systems engineering and discussed the evolution of the HKMF that met the first two requirements and shows promise of meeting the remaining two requirements. In its early days, the HKMF has:

- Provided an understanding of why systems engineers can't agree on their roles and activities (Chapter 18).
- Provided an understanding of the reasons for the overlap between systems engineering and management from the origin of the situation discussed by (Johnson, 1997).
- Shown a relationship between Enterprise Systems Engineering and the Engineering of Systems.

21.14 Conclusions

The conclusions from the research discussed in this book culminating in this Chapter are:

- Although the HKMF does not currently meet all four requirements for a framework for understanding systems engineering, it has been able to provide explanations for some of the problems in today's systems engineering paradigm. It is thus a useful educational tool and provides a baseline stepping stone for future research.
- While the HKMF does not provide a theory of systems engineering, it does provide a platform for further research into the nature of systems engineering.
- By virtue of the HKMF, systems engineering can fulfil Wymore's requirement to become a discipline (Wymore, 1994) since systems engineering now meets Kline's definition of a discipline, namely it has *"a specific area of study, a literature, and a working community of paid scholars and/or paid practitioners*" (Kline, 1995) page 3).
- The HKMF embodies Checkland and Holwell's (A), (F) and (M) (Checkland and Holwell, 1998) and hence can be also considered as a candidate for the first formal representation of research into the discipline of systems engineering. This makes systems engineering a reality not a myth[105]!

[105] Refer back to Chapter 2.

2008

22 Luz: from light to darkness: lessons learned from the solar system[106]

In teaching systems engineering the relationship between functions, physical decomposition and requirements during the process of defining, designing and developing the system, has been difficult to get across to the students. While trying to improve the learning process, an explanation of the relationship between functions, physical decomposition and requirements during the process of defining, designing and developing the system based on a modification of the FRAT views of a system (Mar, 1994) was tried on undergraduate students at UniSA in 2006-2007 with positive results (Kasser, et al., 2007). This Chapter documents the LuZ SEGS-1 system design process in the form of the FRAT views demonstrating the intertwined relationships between requirements, functions and their allocation to components at lower levels of system decomposition. The Chapter also provides examples of alternative choices, discusses them and documents the choices with the reasons for selection. Several lessons learned from the project are also provided.

22.1 Introduction

The systems engineering process contains the sequence of activities in which as a response to an identified need, a conceptual system is defined, designed, developed and placed into service. However, in teaching systems engineering the relationship between functions, physical decomposition and requirements during the process of defining, designing and developing the system, has been difficult to get across to the students.

While trying to improve the learning process, an explanation of the relationship between functions, physical decomposition and requirements during the process of defining, designing and developing the system based on a

[106] The writing of the paper was made possible by a grant from The Leverhulme Trust to Cranfield University.

modification of the FRAT views of a system (Mar, 1994) was tried on undergraduate students at UniSA in 2006-2007. During the learning process, the questions they asked indicated that they were demonstrating a better grasp of the concepts than had been demonstrated by postgraduate students in earlier courses (Kasser, et al., 2007).

This Chapter uses the adapted FRAT as a frame in which to describe the relationship between functions, physical decomposition and requirements using as an example the definition, design and development project for the control and electronics subsystem of the LuZ SEGS-1 system in 1981-1983 (Section 11.8.1). The Chapter also provides some lessons learned from the project.

22.2 The FRAT approach

Functions Requirements Answers and Test (FRAT) (Mar, 1994; Mar and Morais, 2002) was introduced as four views of a system. However, the "A" and "T" in FRAT are also useful for investigation of conceptual alternatives which provides a useful perspective, namely:

- **Functions** which define what functions the solution must perform (operational and functional perspectives).
- **Requirements** which define how well each function must be performed[107] (quantitative perspective).

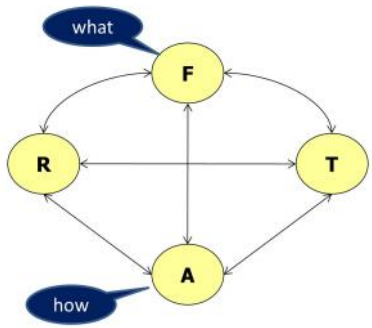


Figure 22-1The FRAT approach

- **Answers** which use the scientific perspective to search not only for one solution, but for a better solution as well and manage risk associated with that solution (descriptive perspectives). The answer element concludes with a decision as to which of the alternative answers identified as providing a solution to the problem is to be selected for implementation.
- **Tests** which demonstrate that the selected solution performs the needed functions according to the requirements. The scientific perspective is used to create the tests and the quantitative perspective to determine the expected test results.

[107] The A paradigm definition see Chapter 28.

In use, FRAT is not performed in a sequential manner. It is instead performed in an iterative series-parallel manner in which each of the four elements is both an input to and an output of the others as shown in Figure 22-1 since the findings in one element may affect the contents of another. The use of FRAT will now be illustrated in the case of the systems development process for the world's first commercial solar electrical power generating system (SEGS-1). FRAT is used at each level of system elaboration and in each subsystem as shown in Figure 22-2 or mapped into the problem-solving view of the Waterfall as shown in Figure 22-3.

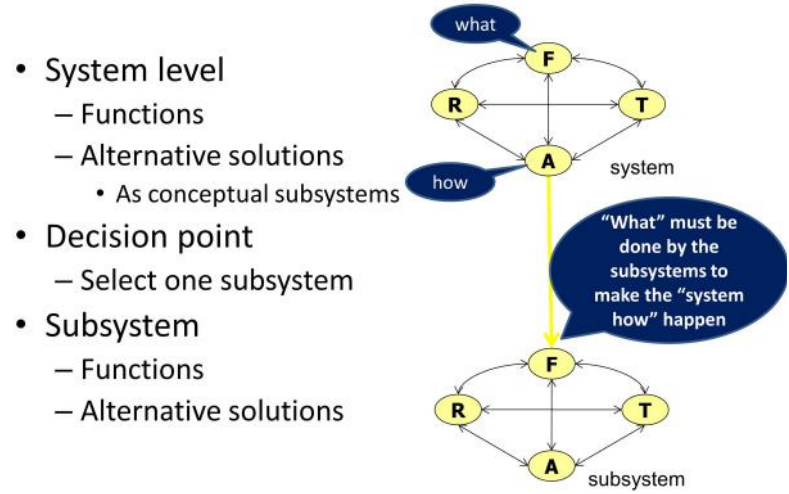


Figure 22-2 FRAT elaborated down the hierarchy

22.3 Background to the case study

The LuZ Group, a start-up joint Israel-American venture defined, designed, developed, installed and operated SEGS-1 in 1981-1983. At the design time, as the first of its kind, SEGS-1 initially only existed as a vague concept and met Donaldson and Siegel's definition of a very high risk project (Donaldson and Siegel, 1997). SEGS-1 was installed in the Mojave Desert in California and the Research and Development was in performed in Jerusalem. SEGS-1 was intended to generate electrical power from the sun by focussing the sun's rays on about 600 parabolic mirror trough reflector collectors each about 40 meters long. The operation of each parabolic trough reflector would be monitored and controlled by a microprocessor based LOC. Each LOC would control a motor that would position the parabolic mirror, receive information about the angle of elevation of the mirror and the temperature of the oil in the pipe positioned at the focus of the trough. Oil would be pumped through the piping, and as long as the LOC would keep the reflector pointed at the sun within an accuracy of ±0.2 degrees, the oil would be

heated. The hot oil would be pumped around the field and into a heat exchanger to generate steam. The steam would then drive a turbine that generated up to 15 Megawatts of electrical power. Although it would be a complicated system, it would still have a conversion efficiency of about 40%, greater than any alternative method of harnessing solar energy at the time.

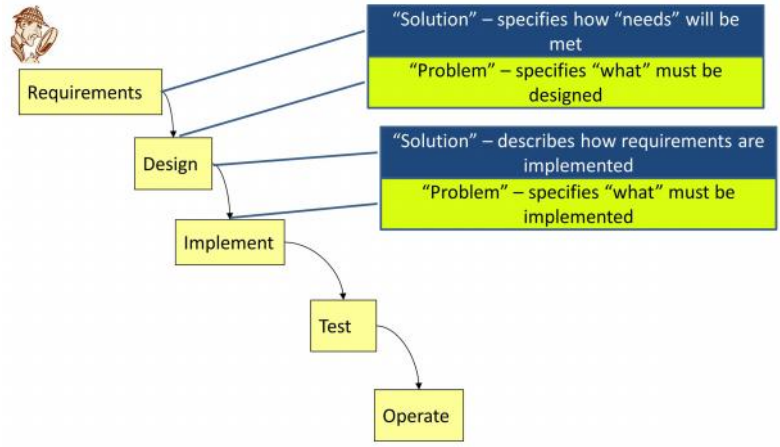


Figure 22-3 Problem-solving view of the waterfall

22.4 At the system level

Consider the system from various perspectives on the perspectives perimeter[108] (Kasser and Mackley, 2008), the details and insights that emerge include[109]:

22.4.1 Big picture perspective

The big picture perspective includes:

- SEGS-1 is the containing or meta-system.
- The system under consideration/discussion is limited to the control and electronics system (CES) for the array of solar collectors within SEGS-1.
- The CES is a system in itself, and at the same time is a subsystem of SEGS-1.
- The oil pumping and heat transfer elements of the solar field, and the array of mirrors, are outside the CES. They interface to the CES elec-

On [108] These perspectives evolved into the Holistic Thinking Perspectives (Kasser, 2013) summarized in section 28.2.

[109] This is not meant to be a complete treatment at the system level.

tronically via the oil temperature sensors and mechanically via the motors.

- The solar collection array is to be located in an area the size of a football field.
- There is an adjacent totally decoupled system, an experimental heliostat solar power generating field which may provide a false sun via the reflection from the central tower and the LOCs may lock on to it and fail to follow the sun.
- There is no space in the development facility to build and test a large array of LOCs prior to deployment half way around the world. The design and test programs will have to compensate for this constraint.

22.4.2 Operational perspective

The CES operates automatically daily as long as there is no major cloud cover. Manual operation takes place in certain scenarios. Each and every mirror can only move in one axis – elevation. The azimuth is fixed approximately north-south.

22.4.3 Functional perspective

The mirrors deploy to track the sun in the morning as soon as the sun is high enough to generate positive power, track the sun during the day and stow in the evening. The CES performs the following functions:

- **Stowed.** The mirrors are at rest in an upside down position. This minimises dirt collection and sand abrasion of their surfaces which reduce the reflection coefficient and hence the efficiency of the system. Since the mirrors also act as radiators when not acting as heaters, stowing the mirror minimises heat loss to outer space.
- **Deploying.** The mirrors are moving up from the stowed position to begin to track (follow) the sun.
- **Tracking.** The mirrors follow the sun across the sky keeping the oil pipe at the focus, allowing the sun to heat the oil.
- **Stowing.** The mirrors are returning to the stow position at the end of the day.
- **Manual movement.** The mirrors are moving as commanded by the operator.
- **Idle.** The mirrors are stationary.
- **Data collection, storage and reporting.** The system generates, stores and reports information about the positions of the mirrors, and temperatures of the oil at their foci.

22.4.4 Structural perspective

The insight from the structural perspective includes that even though there

are about six hundred LOCs, since each LOC will be identical to the others, there are really only four subsystems. These are the central control station (CCS), the LOCs, the interconnecting network between the CCS and the LOCs, and the power distribution system.

22.4.5 Generic perspective

The insights provided by this perspective include:

- The CES architecture is similar to a constellation of low earth orbiting satellites and a control station. This allows spacecraft telemetry tracking and control techniques to be considered as design options.
- The mirror positioning function is similar to a satellite ground station positioning function.
- Short duration loss of insolation due to intermittent clouds is similar to the loss of synchronizing pulses in an analogue TV signal, so a flywheel technique could be employed to compensate.

22.4.6 Continuum perspective

There are at least three design choices,

- One end of the continuum, in which a smart CCS manages the entire array as well as collecting and storing information about the CES,
- the other end of the continuum in which a dumb CCS collects and stores information about the system while the LOCs perform the mirror management functions, and
- a mixture of the previous two, namely somewhere along the continuum.

22.4.7 Temporal perspective

This perspective indicates that the efficiency of SEGS-1 is expected to reduce over time due to physical effects in subsystems adjacent to the CES (such as loss of vacuum in the oil pipes). While SEGS-1 can be maintained after hours, it is undesirable (extra operating cost) and the identification and replacement of a failed component should be quick. It would also be desirable for the CES to be able to predict failing vacuums in the oil pipes so that replacements could be scheduled.

22.4.8 Quantitative perspective

This perspective shows the following issues.

- Feasibility study calculations have shown that each 40 Meter long mirror must be pointed at the sun with an accuracy of ±0.2 degrees.
- The CES uses AC power to move the mirrors and power the LOCs. To be practical, SEGS-1 must put more power into the electrical power grid than it takes out.

- The more power it can produce, the greater the revenue to the investors who will own SEGS-1.

22.4.9 Scientific perspective

Alternative candidate solutions for the CCS are minicomputer or microcomputer based.

22.5 FRAT at the system level

Applying FRAT at the CES system level we have[110]:

22.5.1 Functions

The control function includes the following:

- Deploying the entire array of mirrors when the power to be generated by SEGS-1 is more than the power to be used.
- Tracking when and while the sun shines; idle for periods of cloud cover.
- Stowing the array when the power to be generated by SEGS-1 is less than the power to be used.
- Gathering, displaying and storing status information about the operation of SEGS-1.

22.5.2 Requirements

There are two performance requirements on the CES namely[111]:

1. In operation, SEGS-1 shall generate more power than it uses.
2. SEGS-1 shall generate the maximum possible amount of power each day.

22.5.3 Answers

The key to meeting the requirements is the accuracy of positioning the mirrors. Calculations have shown that the derived requirement to be levied on the mirror pointing function is to point the mirror at the sun with an allowa-

[110] Note there are Meta-system considerations as well in the discussion since they affect the control functions.

[111] These requirements have not been stated in numerical terms. The feasibility study has shown that the system can potentially generate 15 Megawatts of power. However, the power it uses to do that will be a function of the motors, the power distribution losses, etc. That were unknown at the time the requirements were stated. Stating them as goals in this manner means that once the design has been made, the 'deploy' and 'stow' commands can be based on calculations or on the amount of solar insolation.

ble offset of only ±0.2 degrees. The following alternative solution options for meeting this requirement were considered:

- A tight control loop which would depend on accurate system information (tight tolerances) about the longitude, latitude, and mirror alignment with respect to north of the field; angle of sun sensor with respect to mirror axis, and other pertinent parameters for each mirror.
- A sloppy control loop which would not require the tight tolerances of the previous option, with self-regulating components to maintain the system in the steady tracking state.

Some of the factors affecting the choice were as follows[112]:

- While one LOC had the capability to control more than one mirror, the system architecture was simpler if each LOC only had to interface to a single mirror. In addition, the embedded software would have fewer instructions.
- The amount of functionality in the LOC microprocessor was yet to be determined.
- The sun seems to travel across the sky at a rate of 15 degrees an hour, or one degree in four minutes. The mirrors move slowly enough that the position control algorithm could be located in either the LOCs or in the central station.
- The quantitative perspective shows that assuming that each sun-sensor has a slightly different offset when installed on its mirror, if the position control algorithm (CES function) is located in the CCS (physical component), it will have to compute the desired elevation angles for each mirror in the entire array and update the LOCs within 24 seconds (requirement on CCS). This means that the CCS and LOC processor software cycle times, the network data rates, message lengths and other parameters will have to be determined to ensure that the CES can meet the performance requirement.
- If the position control algorithm (CES function) is located in the LOC (physical component) number of instructions in the software will have to be estimated to ensure that the microprocessor cycle time is fast enough (requirement on LOC) to perform the computations.
- The angle between the array of mirrors and due north is difficult to determine to decimal point accuracy.

[112] There were other decisions to be made that have not been included in this extract. One such decision early in the system design phase was should one LOC control a single mirror or should it control several mirrors? And if so, how many?

The functional analysis was first performed for a single function in the CCS controlling all the LOCs and as shown below was found to be problematic with the microprocessor solution. However, instead of choosing the minicomputer alternative, an innovative partitioning approach was then postulated (scientific perspective) coming from the generic perspective on the similarity between the CES and a constellation of satellites. This physical allocation split the functionality between the LOC and the CCS (with minimal coupling) enabling the microprocessor based CCS solution.

The design decision was to develop a physical architecture in which one LOC controlled one mirror, and use the sloppy algorithm using a self-regulating approach; partitioning the control system functionality such that the responsibility for making the decision to deploy or stow resided in the CCS, but the position control function of each mirror was the responsibility of its LOC.

This approach eliminated many of the calculations on the data rates and tolerances that would otherwise have been required. The only serious technical risk associated with this choice was that the chosen microprocessor cycle time might not be sufficient.

22.5.4 Test

Previous experience with microprocessor-based satellite ground station antenna controllers indicated that this risk had a very low probability of occurrence. Subsequent breadboarding verified that the LOC could do the task.

22.6 FRAT at the subsystem level

The functionality and requirements for the CES are now expanded at the subsystem level. This process contains steps in which options are determined and choices made. Sometimes trade-offs have to be made between the locations of functionality, that is, to which part of which subsystem the functionality to meet a specific CES requirement is allocated.

A selection of the various physical choices to be made in allocating CES functionality to the physical subsystems, namely designing the CCS, network and power distribution subsystems is shown in Figure 22-4[113]. These choices were:

1. **CCS.** The physical alternatives were to use a mini- or microcomputer.
2. **The Network.** The physical alternatives were to use an Ethernet or twisted pair shielded wire.

[113] An alternative depiction is a decision tree.

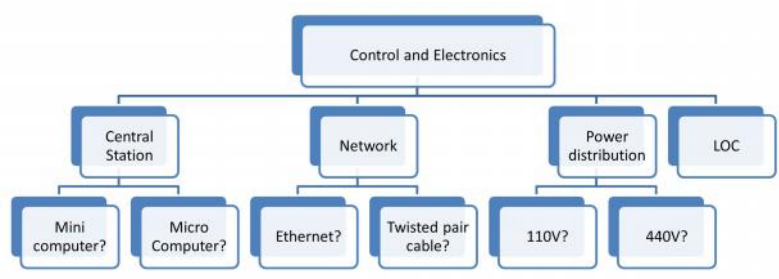


Figure 22-4 Part of the design choices at the subsystem level

3. **The Power distribution.** The alternatives were to distribute power to the LOCs at 110V, 240V or 440V using single or three phases.
4. **The LOC.** The LOC was the most complex subsystem. Some of the physical alternatives are shown separately in Figure 22-5 to keep the figures simple.

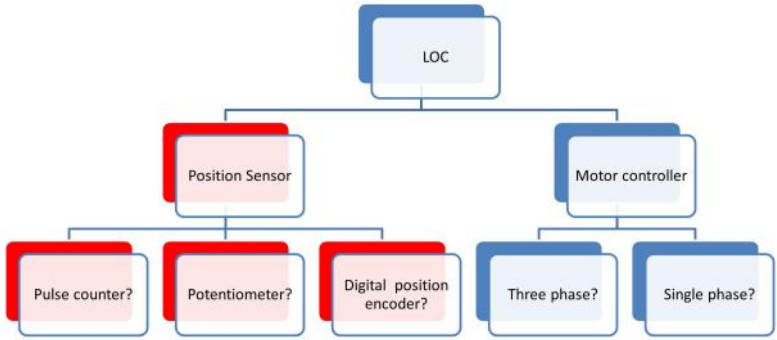


Figure 22-5 Part of the design choices for the LOC

Consider the application of FRAT to the network, CCS, power distribution and LOC subsystems in making those choices.

22.6.1 FRAT applied to the CCS

Applying FRAT at the subsystem level to the CCS we had:

22.6.1.1 Functions

The functions include:

- Automatically deploying and stowing the LOCs at the appropriate times of day.
- Acting as an operator interface to the array of mirrors allowing the operator to command and interrogate the LOCs.
- Displaying CES status information.

- Storing CES status information on a 10 Megabyte hard drive, offloading the data on magnetic media for archival storage offsite should the function be desired by the system developer or customer in the future[114].

22.6.1.2 Requirements

The requirements inherited from the CES level included:

- Determination of sun elevation angle for the field throughout the day.
- Commanding and interrogating the LOCs individually or en-masse.
- Displaying CES status information in a user friendly manner (location on the control screen, font size, colour, etc.)
- Sounding alarms when various conditions exist.

Requirements on the CCS to display information in various colours depending on the states of the CES and sounding the alarms are ergonomic requirements, not functional and performance. They are inherited from the class of system to which the CCS belongs.

There were no detailed requirements for the actual operator commands and display of information. This functionality evolved in a rapid prototyping environment; the designers and systems engineer evolved the command and displays starting with a concept of "what and how" and working with the software to modify it as experience was gained. Ideas from the adjacent Heliostat system were considered and included as appropriate. The requirements were written down eventually for the purpose of documenting the actual functionality achieved rather than documenting what functionality would have to be achieved[115].

22.6.1.3 Answers

The choices of design for the physical allocation of the CCS functionality were to use a minicomputer equivalent to the Hewlett Packard 3000, the Apple Lisa and a Z-80 based S-100 microcomputer[116]. However the sloppy control split function approach reduced the workload of the CS to the point where a Z-80 based S-100 Bus microcomputer[117] could do the job, saving the project around $300,000 for each of the three CCS that would be needed;

---

[114] The hardware constraints were included in the functional analysis because that was the state of the art at the time.

[115] Considering the situation with hindsight, it would have been more appropriate to write a test procedure that confirmed the functionality rather than document the functionality in a traditional requirements document.

[116] The IBM PC wasn't even a gleam in some engineer's eye at that time, and the Apple Lisa was a not very serious candidate.

[117] With a new colour display card to be designed since one was not available as COTS.

namely $900,000[118]. Given that the design team had experience programming the Z-80[119], and the need for cost saving (important to a start-up company), the design choice was to go with the Z-80 microcomputer.

The CCS software was portioned into Builds using the cataract development methodology (Chapter 13). The Builds were designed to increase automated functionality as more was learned about the operation of the system. Remember this was a 'first of its kind' system. Thus Build 0 would provide the basic operator interface functionality allowing the system to be operated and tested manually. Build 1 would provide the initial automated functionality and future builds would incorporate additional functionally, transferring functions commonly performed by the operator into automatic sequences and any other approved change requests. Since delivery to the customer was dependent on the dates of installation of the mechanical and thermal adjacent SEGS-1 subsystems and not on the date of completion of the software, this approach mitigated the risk of schedule slips due to software related problems and ensured that the CES would perform its basic functionality when delivered. As it happened, the CES was delivered with Build 1 fully operational.

22.6.1.4 Test

The design decision was supported by experience and calculations that the Z-80 microcomputer was fast enough to do the job.

22.6.2 FRAT applied to the network subsystem

Consider the application of FRAT to the network, subsystem.

22.6.2.1 Functions

The functions were the communication between the CCS and LOCs.

22.6.2.2 Requirements

The requirements included:

- Low data rates between the central station and the LOC due to the sloppy algorithm and slow rate of change of both the sun angle and mirror movement
- Operate no matter what the electromagnetic interference (EMI) environment. The degree of EMI was unknown; there could be thunderstorm induced transients or induction of signals into the underground cable network from nearby local radio stations. As such, from the quan-

[118] One for software development, one for operations in the field and one as an operational spare.
[119] In the form of the Intel 8080.

titative STP there was little point in putting any numbers into the requirements.

22.6.2.3 Answers

The alternative physical design choices to perform the functions to the requirements were coaxial cable-based Ethernet, then in its infancy and hence high risk, or a low data rate twisted pair shield cable, a lower risk option. The low data rates meant that the design need not use Ethernet, nor did it need to compensate for signal deterioration in long coaxial cable runs. There was also a choice in which type of communications protocol to use, the choices being between a polled approach in which the CCS interrogated each LOC in turn sequentially and some kind of random access protocol. This choice was independent of the physical design. The design decision allocating the functions to physical components was to use twisted shielded pair cable. In addition, the communications was implemented using:

- A polling protocol in which the central station polled each LOC in turn.
- The data format was short commands and responses in ASCII at 1200 Baud.

22.6.2.4 Test

The decisions were verified by a calculation that showed that 1200 Baud was fast enough that the CCS could poll and receive a response from each LOC within the predetermined minimum cycle time[120]. This design (functions to physical allocation) approach also allowed a then-common hand-held ASCII data terminal to be used to troubleshoot communications problems because the technician could bridge the cable and monitor network traffic, or as appropriate, disconnect the network cable and transmit signals to a LOC or the central station and see the response on the hand display.

22.6.3 FRAT applied to power distribution

Applying FRAT to the power distribution subsystem we had:

22.6.3.1 Functions

The distribution of power to the array of motors and LOCs.

22.6.3.2 Requirements

The requirements included:

- The power distribution subsystem shall distribute AC power to the LOCs.

[120] Mitigating this risk was the fact that the rate could have been lowered to 300 Baud which would still have been fast enough to provide status information. However, it would have been slow from an operator's point of view.

22.6.3.3 Answers

The physical implementation choices were to distribute power to the LOCs at 110V, 240V or 440V using single or three-phase power. Factors affecting the choice included the impedance losses in the power cables - the higher the voltage, the lower the losses. The design methodology employed was a problem avoidance approach. Early prototyping of a 240V three-phase approach had showed that the silicon controlled rectifiers had a high failure rate – a high-risk situation. Instead of investigating the causes of these failures, we used continuum thinking to adopt an alternative inherently low risk mixed voltage distribution approach as follows. The bulk of the power would be distributed at 440V into the field to strategically located power distribution units (PDU). The loading on the supply phases was balanced by using different phases in different parts of the field. The PDUs would contain a step down 440/110V isolation transformer and transient suppressor. Power from the PDUs would be distributed to a bank of LOCs at 110V (±10%). This design choice had the following benefits.

- The use of 110V single phase motors which were testable in Israel using a standard 240/110 V step down transformer.
- The step down transformers in the PDUs also acted as high frequency transient chokes.
- The 110V distribution allowed a single power line to be routed to the LOCS for supplying power to both the motors and the LOC internal electronics.

22.6.3.4 Test

The feasibility of the decision was verified by analysis at the design phase and subsequent operation at the acceptance test[121].

22.7 The LOC subsystem

The LOC became the heart of the system and is discussed more thoroughly than the previous subsystems in this paper. Insights and observations from considering the LOC from various perspectives include:

[121] Would further detailed analyses have changed the geometry of the power distribution and perhaps resulted in lower cable impedance losses? Perhaps. However the customer was satisfied with the architecture and the system was deployed to schedule, so the solution was a correct one. This does not mean that these detailed analyses should not be made sometime in the future when resources become available, and the results of the analyses be used in the design of follow-on systems.

22.7.1 Operational perspective

Each LOC operates in a self-regulating automatic manner such that the mirror it is controlling, is always pointing at the sun. The operator at the CCS or the automatic function in the CCS commands and interrogates the LOC in accordance with the operating scenarios alluded to in the system level operational perspective (not provided in this Chapter).

22.7.2 Functional perspective

Each LOC performs the following functions:

- **Deploy.** Upon receipt of a Deploy command, deploys the mirror to the commanded position. Note the position has been determined by the CCS to be slightly lower than the calculated sun elevation angle.
- **Search.** Upon reaching the deploy position, moves the mirror forwards a few degrees until the sun sensor acquires the sun.
- **Track.** Once the sun sensor has acquired the sun, moves the mirror forwards past the sun, stops and waits for the sun to catch up. When the sun catches up, moves it forwards past the sun again and waits, and so on until evening. This function is self-regulating and is independent of the communication link to the CCS.
- **Loss of sun.** Flywheels as if it was tracking for a short period of time. This function came from the generic perspective; the length of the 'short time' came from the quantitative perspective.
- **Idle.** Does not move until told otherwise.
- **Command and control.** Moves the mirror and provides oil temperature, and functional mode (deploy, search, track, loss of sun and idle) and position information to the CCS upon receipt of the appropriate commands.

22.7.3 Big picture perspective

The LOC subsystem interfaces to a sun sensor, a position sensor and the motor on the adjacent mechanical mirror system as well as a temperature senor in the adjacent heat flow system.

The perspective also points out the risk potential for a mixture of various problems with either adjacent systems or internal subsystems as follows:

- Incorrect interfaces since the mirrors would be coming from Germany.
- Incorrect interfaces since the position sensors would be subcontracted to a Japanese manufacturer.
- The LOCs would be integrated in Jerusalem, from parts purchased both in Jerusalem and California.

- Cabinets and housings were custom made in Israel. Manufacturing tolerances were loose so that mechanical parts were not always interchangeable.
- Power transformers were made in Tel Aviv. The quality as it turned out was excellent.
- Undesired emergent properties. There was no space to build, store and test a full array of more than 600 LOCs in Jerusalem. Anticipating undesired emergent properties to appear after installation in the field[122], the LOC firmware would be in EPROMS to allow for speedy upgrades. Consequently, there was a configuration risk of not having future field-installed upgrades in-place, and subsequent reporting of already fixed problems.
- The LOCs would contain unique identifiers in chip headers. Each would have to be installed in the designated LOC as part of the system integration in the field. If the headers were to be installed in the wrong LOCs, the CCS would experience problems.
- The communication cables would be purchased in the USA, manufactured in California and shipped to the site. This provided the only problem[123] experienced in the CES (Section 11.8.1). The cables were fitted with push-on connectors for fast and error-free connection. However, the cables used in Jerusalem were flexible and push-on connections stayed in place. The cables purchased in California were not identical and were less flexible. Slowly over time the tension in the cable caused it to pull away from its printed circuit board mounted socket. However, once the cause was determined, the fix was obvious[124].

22.7.4 Structural perspective

The LOC architecture was that of a standard embedded digital microprocessor with analogue interfaces to the sun sensor, oil temperature sensor, a digital interface to the mirror elevation angle sensor, a serial balanced digital driver interface to the network, and a high-voltage control interface to the motor control circuits. The LOC also contained a COTS alternating current (AC) power supply.

---

[122] These emergent properties would be a property of the large array and would not be seen in development.

[123] The sun sensor was produced by the Physics department who were also responsible for the heat flow subsystem, so technically, the sun sensor problem in section 11.8.1 was a heat flow subsystem problem

[124] Tie it down so it wouldn't lift off.

22.7.5 Generic perspective

The LOC is an electronic device in an outdoor desert environment. As such it inherits environmental requirements from that domain.

22.7.6 Continuum perspective

The LOC is not just part of one system, it can be considered as a subsystem of two different systems[125], namely:

1. The CES, and
2. The mirror system which contains the LOC, the mirror, the motor that moves the mirror, the mirror elevation angle position sensor, the heat flow elements at the focus of the mirror and the oils temperature sensor in the heat flow element at the centre of the mirror.

In addition, subsystem boundaries can be drawn in various ways depending on the interest, for example, the mirrors are part of the SEGS-1 mechanical subsystem, and the heat flow elements are part of the SEGS-1 heat flow subsystem.

22.7.7 Temporal perspective

The LOC need not operate 24 hours per day seven days per week since the solar array did not generate power at night. However, failures should be readily apparent to the operator so that replacements could be made speedily. Down time is loss of revenue time.

22.7.8 Quantitative perspective

Sun movement is relatively slow and is measured in degrees of arc/minutes. The rate of collecting telemetry information from the LOCS can be just as slow. Thus exact data rates need not be specified at the system conceptual design review. The test function at the subsystem level will later show that the data rate selected would be/was fast enough for this situation.

22.7.9 Applying FRAT to the LOC

Applying FRAT at the subsystem level to the LOCs we have:

[125] Note that during the design process both views were employed at different times for different purposes.

22.7.9.1 Functions

The functions are as described in the functional and performance perspective above.

22.7.9.2 Requirements

Some of the LOC requirements are:

- The mirror position accuracy requirement is inherited from the CES level requirement, namely ±0.2 degrees.
- The environmental requirements are inherited from the standard outdoor desert requirements.
- The data transfer rate for command and control between the CCS and the LOC are low due to the sloppy algorithm for controlling the LOC pointing angle.

22.7.9.3 Answers

Some of the design options for physically implementing the LOC functionality are shown in Figure 22-5. Design decisions at this level were made between alternative sensors namely:

- **Position sensor.** The function is sensing the position of the mirror, the physical choice was between:

  1. some sort of pulse counter which would count revolutions of the motor from the stow potion and compute the angle reached,
  2. an analogue potentiometer in which the resistance would be a function of the angle of elevation and
  3. a digital sensor which provided position information in Grey code.

  Each option was considered as summarized herein.

- **The pulse counter** needed some mechanical construction and electronic circuitry and was relatively low risk. The cost was unknown.
- **The potentiometer** was open to problems in the future due to wear and tear on the track, since the same section would be traversed daily. There was no information on how readily available components behaved under these conditions. The use of high reliability potentiometers from missiles was considered, but their operating conditions were very different since a missile spends most of its time in storage and just a few minutes in flight. In addition the interface would have to be individually calibrated for each sensor in the field after installation. From the temporal perspective, if the sensor degraded in use to the point of replacement, the replacement was expected to be costly and since they could all be expected to fail within the same time frame, the system would have to operate in a degraded mode for some time, while the replacements were manufactured, shipped and installed.

- **The optical sensor** option was the simplest. The Grey code provided a simple unambiguous position with a digital interface.

The problem of coupling the position sensor to the mirror should the pulse counter not be chosen, was solved by "Rogo", the head of the Mechanical Department who conceptualized hanging a weight on the position encoder shaft so that it acted as a pendulum. Movement of the mirror would change the angle between the pendulum and the fixed point on the mirror. This design could be used with both the potentiometer and the optical sensor. The pendulum and position encoder were encapsulated in an environment proof container that could easily be interfaced to the mirror.

The cost of the selected option was $300 per unit, which was not cheap when the total number to be purchased was 600 or so. However, the simplicity of the design, coupled with the low risk made it the obvious choice at the time. Since the department resources and schedule were limited and an investigation of the alternative choices would take time with no guarantee of lower cost (costs of the alternative sensors and signal processing components, not to mention assembly costs), the optical sensor option was chosen and a contract awarded for a few prototypes, and once they had been tested, contract for the manufacture and shipping of the remainder to the installation site was awarded[126].

- **Sun sensor.** The sensor was a standard two-diode sensor shown in Figure 11-4, but there was no specification correlating the incident light to the diode characteristics. This was a major risk because pointing accuracy had a tight tolerance and was critical to the success of the system and there were also no vibration specifications for the mirrors as a result of movement or for any other cause. A "specification free design" was bread boarded, tested and put into service, but there was no firm information as to the design margin.

22.7.9.4 Test

The sun sensor electronic interface was tested in the laboratory and the embedded software shown to work under all test conditions so the system was passed for deployment. Mind you, as it turned out there was a preventable problem with the sun sensors (Section 11.8.1). The sun sensor used a lens to focus the sun onto the pair of photo diodes. During the assembly process the diodes were glued to a base plate with transparent glue. The physics department who were building the sun sensors did not place a manufacturing process requirement that there be no glue on the side of the diode illu-

[126] Had the quantity to be manufactured been much greater, a more detailed design and cost analysis might have had to be done to determine the lowest total cost.

minated by the sun. After all, the glue was transparent. A year or so later, they found that the glue slowly became opaque when subjected daily to the very high temperature at the focal point of the lens. This phenomenon resulted in the need to replace all the sun sensors. From a manufacturing perspective, there was little difference in mounting the diodes if the glue could or could not be allowed to cover the face of the diode, just a matter of care and a few extra minutes of time. Nobody in the physics department which designed and produced the sun sensors employed the temporal perspective to ask about possible changes to the characteristics of the glue over long periods of time under high temperature. If the requirement had been placed on the process, not to allow glue on the face of the diode, the characteristics of the glue under the high temperature conditions would not have mattered and the expensive sun-sensor replacements would have been avoided. This is an example of introducing an unnecessary failure mode by not utilizing the "don't cares" due to the lack of system thinking. Thus the lesson learned is that if it doesn't make any difference don't do it.

22.8 Discussion on the use of FRAT

The use of the modified FRAT concept has shown how requirements, functional analyses and physical allocation flow from the system to subsystem as the design progresses. One example is that of mirror positioning. The system requirement was to generate power from the sun. This translated to moving and positioning function and a pointing accuracy derived requirement of ±0.2 degrees. Further functional decomposition as part of the design process split the moving and positioning function between two physical subsystems the CCS and the LOC. The CCS functionality determined when to deploy and stow the mirrors, while the LOC functionality kept it positioned to meet the performance requirement. The pointing accuracy requirement was thus implemented in the LOC. Further decisions as to what type of physical element would perform the mirror position sensing function were also shown.

22.9 Lessons learned from the project

This last part of the Chapter contains some of the lessons learned from this project. A number of instances are already documented and references are provided in those instances. The lessons learned from this project include:

- KISSE or keep it simple, systems engineer!
- Focus on people not on process.
- Keep the number of requirements small and pertinent to the mission.
- Requirements are only a communications tool.

- Functional decomposition creates physical architecture and drives design.
- Avoid analysis paralysis.
- View the problem from several perspectives or frames.
- Current 'don't cares' can hurt you in the future.
- Design for test and maintenance.
- Reuse of components simplifies system software as well as hardware.

Consider each in turn.

22.9.1 KISSE or keep it simple, systems engineer!

Was this a system, or a system of systems? System boundaries are in the eye of the beholder (Churchman, 1979) page 91). A LOC was a system in itself containing a mixture of COTS and custom components; the whole array of LOCs was a system. On the other hand, the array of LOCs, communications network and the Central Station was a system. Not only that, but a single LOC and the sensors mounted on its mirror also comprised a system and the whole field was another system. How many systems actually were there? The answer to the two questions is "it depends". System boundaries are drawn for a purpose and represent an abstraction of part of the real world for some purpose. They should be used appropriately.

22.9.2 Focus on people not on process

The expertise must be intuitive in the person, not in the manual or in the Standard in other words, it must be "*a philosophy and a way of life*" (Hitchins, 1998). A process standard would probably have killed the project. Note in the discussions above, how trade-off analysis for choosing physical options to meet required functionality were completed only to the point of determining the feasibility of the option, and not down to crossing the last 't' and dotting the last 'i'. This was because the systems engineer knew when to stop analysing and start deciding. This lesson supports the statement by Robert Frosh, when he was Assistant Secretary to the US Navy who stated: "*Systems, even very large systems, are not developed by the tools of systems engineering, but only by the engineers using the tools*" (Frosch, 1969).

22.9.3 Keep the number of requirements small and pertinent to the mission

There were only two top-level requirements. If there is consensus on what the system is to do, don't specify details that can best be left to designers and other stakeholders to determine interactively further down the lifecycle.

22.9.4 Requirements are only a communications tool

If there is a clear understanding of the purpose of the system you may not need requirements (Chapter 16). The engineers and technicians at LuZ spoke various combinations of Russian, English, French, Romanian and Hebrew, because most were immigrants and there were times when there was no common language in a meeting. Yet in spite of these communications difficulties, and the lack of written requirements, we knew what we were building and the project succeeded[127].

22.9.5 Functional decomposition creates physical architecture and drives design

Consider alternate decompositions at design time. The rules for functional decomposition are (Section 18.9):

- Minimize coupling and maximize the cohesion.
- Consider the operator as part of the system.
- Use self-regulating subsystems.
- Use railroad buffers for signal passing.

Note use cases and concepts of operations describe functions. For example, we considered various functional options for the mirror positioning functionality and exposed the advantages and disadvantages of each option. The eventual choice was to separate the sun elevation angle calculation function from the mirror pointing functions. This allowed one function to be located in the CCS and the other in the LOC. Since the mirror pointing function was designed as a self-regulating function, there was minimal coupling between the two functions (i.e. just the command to deploy, and the necessary angle). It also allowed us to complete the software ahead of schedule.

22.9.6 Avoid analysis paralysis

There is no need to continue an analysis past the point of identification of non-feasibility.

22.9.7 View the problem from several perspectives or frames

Use generic thinking. Someone out there has probably already solved a part of the problem or even your entire problem. This is the core concept in TRIZ (Theory of Inventive Problem Solving) in which it is stated as "*Somebody someplace has already solved this problem (or one very similar to it.) Creativity is now finding that solution and adapting it to this particular problem"* (Barry, et al., 2007). However, you may have to redefine your problem to

[127] Delivered on time within budget and performed to requirements.

use that solution to avoid unnecessary complicating the situation. The generic perspective helps to identify candidate solutions for pattern matching to your situation.

22.9.8 Current 'don't cares' can hurt you in the future

Don't add unnecessary parts even if they appear not to make a difference at design time. The sun sensor glue was an unnecessary part when placed in front of the photo diode as discussed above.

22.9.9 Design for test and maintenance

Test and integration needs to be considered at production process design time as well as product design time. Add test points to both the hardware and software. Embedded software in-circuit emulation will not find all problems. Choose a software architecture such as a 'state machine' which can be thoroughly tested and use buffers to store data being passed between functions for simplifying testing.

22.9.10 Reuse of components simplifies system software as well as hardware

Reuse of the mirror-pointing algorithm in each LOC simplified the CES design. This was an object-oriented approach inherited from hardware expertise.

22.10 Meta-lessons learned

Now just because these lessons were learned from this system, they should not be allied indiscriminately to all future systems. Note, there are two kinds of meta-lessons to be learnt,

- Some of these lessons apply to all projects, and
- Some of these lessons apply to some projects.

Think before you apply them, and ask yourself the following two questions.

- Why did it work here?
- What is different about your project?

22.11 Summary

This Chapter has documented the LuZ SEGS-1 system design process in the form of the FRAT views demonstrating the intertwined relationships between requirements, functions and their allocation to components at lower levels of system decomposition. The Chapter provided examples of alterna-

tive choices, discussed them and documented the choice with the reasons for selection. Several lessons learned from the project were also provided.

2009

23 Reengineering systems engineering

This Chapter documents:

1. using systems engineering in the early phases of the SDLC to develop a conceptual system – the system being developed is a SEBoK and
2. the findings and opportunities generated in those early phases.

The approach was based on identifying activities specific to systems engineering, as opposed to the broad raft of activities that systems engineers might undertake, according to their role. An activity-based definition of systems engineering vs. non-systems engineering role-based definition was developed.

The second part of the Chapter identifies five types of systems engineers, discusses the evolution of systems engineering in terms of those five types, and hypothesizes that a major cause of the failure of systems engineering is the allocation of inappropriate types of systems engineers to early lifecycle phase systems engineering activities. The Chapter concludes with some insights and recommendations for further study.

23.1 Evolution of the role of systems engineering

Descriptions of systems engineering currently comprise different interpretations of the activities known as systems engineering and the broad raft of activities that systems engineers might undertake according to their role in the workplace. This quagmire has developed because different users of the term 'systems engineering' for almost 50 years have chosen or perceived different meanings. For example, one comment from 1960 was "*Despite the difficulties of finding a universally accepted definition of systems engineer-*

ing[128]*, it is fair to say that the systems engineer is the man who is generally responsible for the over-all planning, design, testing, and production of today's automatic and semi-automatic systems*" (Chapanis, 1960) page 357). Jenkins expanded that comment into the following 12 roles of the systems engineer (Jenkins, 1969) page 164):

1. He tries to distinguish the wood from the trees – what's it all about?
2. He stimulates discussion about objectives – obtains agreement about objectives.
3. He communicates the finally agreed objectives to all concerned so that their co-operation can be relied upon.
4. He always takes an overall view of the project and sees that techniques are used sensibly.
5. By his overall approach, he ties together the various specializations needed for model building.
6. He decides carefully when an activity stops.
7. He asks for more work to be done in areas which are sensitive to cost.
8. He challenges the assumptions on which the optimization is based.
9. He sees that the project is planned to a schedule, that priorities are decided, tasks allocated, and above all that the project is finished on time.
10. He takes great pains to explain carefully what the systems project has achieved, and presents a well-argued and well-documented case for implementation.
11. He ensures that the users of the operational system are properly briefed and well trained.
12. He makes a thorough retrospective analysis of systems performance.

Seven of these roles of the systems engineer (activities performed by a person with the title systems engineer) overlap the role of the project manager (activities performed by a person with the title project manager). Research into the reason for the overlapping of the disciplines turned up information as to how the overlap originated in the form of the following statement. *"Driven by cold war pressures to develop new military systems rapidly, operations research, systems engineering, and project management resulted from a growing recognition by scientists, engineers and managers that technological systems had grown too complex for traditional methods of management and development"* (Johnson, 1997). Thus systems engineering, project management and operations research can be seen as three solutions

[128] Fifty years later, nothing has changed in that respect.

to the problems posed by complex systems in the Cold War by three different communities of practice (Johnson, 1997) that have continued to evolve and overlap. Some of the evolution in systems engineering can be seen in the very little overlap between the 12 roles documented by Jenkins and the following 12 systems engineering roles documented by Sheard (Sheard, 1996):

1. **Requirements Owner (RO) Role.** Requirements Owner/requirements manager, allocator, and maintainer/specifications writer or owner/developer of functional architecture/developer of system and subsystem requirements from customer needs.
2. **System Designer (SD) Role.** System Designer/owner of "system" product/chief engineer/system architect/developer of design architecture/specialty engineer (some, such as human-computer interface designers)/"keepers of the holy vision" (Boehm, 1994).
3. **System Analyst (SA) Role.** System Analyst/performance modeler/keeper of technical budgets/system modeler and simulator/risk modeler/specialty engineer (some, such as electromagnetic compatibility analysts).
4. **Validation and Verification (VV) Role**. Validation and Verification engineer/test planner/owner of system test program/system selloff engineer. VV engineers plan and implement the system
5. **Logistics and Operations (LO) Role.** Logistics, Operations, maintenance, and disposal engineer/developer of users' manuals and operator training materials.
6. **Glue (G) Role.** Owner of "Glue" among subsystems/system integrator/owner of internal interfaces/seeker of issues that fall "in the cracks"/risk identifier/"technical conscience of the program".
7. **Customer Interface (CI) Role.** Customer Interface/customer advocate/customer surrogate/customer contact.
8. **Technical Manager (TM) Role.** Technical Manager/planner, scheduler, and tracker of technical tasks/ owner of risk management plan/product manager/product engineer.
9. **Information Manager (IM) Role.** Information Manager (including configuration management, data management, and metrics).
10. **Process Engineer (PE) Role.** Process engineer/business process reengineer/business analyst/owner of the systems engineering process.
11. **Coordinator (CO) Role.** Coordinator of the disciplines/tiger team head/head of integrated product teams (IPTs)/system issue resolver.
12. **"Classified Ads Systems Engineering" (CA) Role.** This role was added to the first eleven in response to frustration encountered when

scanning the classified ads, looking for the INCOSE-type of systems engineering jobs.

Jenkins' roles relate to conceiving and planning the solution system while almost 30 years later, few of Sheard's roles address the original systems engineering approach to conceiving and planning the solution system. Sheard's set of roles relate to interpersonal relationships between the practitioners of disparate skills and disciplines implementing the solution system. Furthermore, according to both Jenkins and Sheard the role of the systems engineer (the activities performed by a person with the title systems engineer) overlaps activities performed (the roles) by people from other professions[129]; the literature provides a wealth of examples of the different overlaps between systems engineering and project management (DSMC, 1996; Brekka, et al., 1994; Roe, 1995; Sheard, 1996; Johnson, 1997; Watts and Mar, 1997; Bottomly, et al., 1998; Kasser, 1996; 2002d) and Figure 23-1. Note the Defense Systems Management College definition of systems engineering as "*The management function which controls the total system development effort for the purpose of achieving an optimum balance of all system elements. It is a process which transforms an operational need into a description of system parameters and integrates those parameters to optimise the overall system effectiveness*" (DSMC, 1996). Notice the use of the term "*management function*"! In addition, see Emes et al. for overlaps between systems engineering and other disciplines (Emes, et al., 2005) and Hari et al. for an example of the activities performed in new product design that overlap systems engineering (Hari, et al., 2004). In addition, the activities performed by the systems engineer in one organisation are different to those performed by a systems engineer in another organization and so are the knowledge requirements for their activities. Consequently, defining a body of knowledge for systems engineering poses a major challenge.

Defining a body of knowledge based on the role of a systems engineer will be difficult if not impossible because the role of the systems engineer has evolved over time so that it is different in practically every organisation. As such, the solution to the problem of defining a body of knowledge for systems engineering is to dissolve the problem by making a change in the paradigm. This approach, which redesigns the system containing the problem or changes the perspective from which the problem is viewed to produce an innovative solution is one of the four ways to tackle a problem (Ackoff, 1999) page 115). The paradigm change is made by making a distinction between a set of activities known as systems engineering and the role of the systems engineer which is the sum of the systems engineering and non-systems engineering activities systems engineers perform in the workplace.

[129] A different set, as seen across the years.

The focus on activities is a return to Hall's definition of "systems engineering as a function[130] not what a group does" (Hall, 1962) page 11) and means that the knowledge needed by systems engineers in their roles will be more than the activities to be defined as 'systems engineering' (see below), and that knowledge can be separated into 'systems engineering' and 'everything else'. The 'systems engineering' knowledge will be placed in the SEBoK, and the subset of knowledge of 'everything else' that will be needed will be out of scope of the SEBoK but will be referenced appropriately.

23.2 Separating out the systems engineering knowledge

In the activity paradigm, various people in various disciplines at various times perform a set of activities from the time a problem is being defined, though the conceptualisation, design, construction and operation of the system that solves, resolves or dissolves the problem to the time that the system has been taken out of service and disposed. This set of activities may be partitioned into subsets in various ways such as by professional discipline (project/engineering management, systems engineering, engineering, new product design, etc.) by time (the phases in the system lifecycle) or by the three streams shown in Figure 2-2. Various systems engineers and non-systems engineers perform different subsets of systems engineering activities and different subsets of non-systems engineering activities. The mapping of the role of the systems engineer to activities is different in different organisations, hence the aforementioned difference in their descriptions when systems engineers get together and discuss their roles[131].

Looking at the structure of organisations from the temporal perspective, in general the structure of organisations is still based on the work of F. W. Taylor who systems engineered his mining organisation and split the work into two streams of activities which have become known as 'management' and 'labour' (Taylor, 1911). However, since that time, the structure of companies and the nature of work have changed. Organizational structures have become flatter, decision making has become decentralized, information is widely shared, workers form project teams, even across organizations, and work arrangements are flexible (Microsoft, 2008b). Taylor's split is no longer applicable. Consequently, this Chapter proposes to reengineer (Hammer and Champy, 1993) Taylor's split for organisations developing systems by splitting work into two different streams, systems engineering and non-systems engineering and further partitioning the non-systems engineering streams as described below.

---

[130] A function is an activity.
[131] Chapter 19.

The INCOSE Fellows definition of systems engineering was considered as a starting point for determining what went into the systems engineering stream. The definition is "*Systems Engineering is an engineering discipline whose responsibility is creating and executing an interdisciplinary process to ensure that the customer and stakeholder's needs are satisfied in a high quality, trustworthy, cost efficient and schedule compliant manner throughout a system's entire lifecycle*" (INCOSE Fellows, 2009). However, if the words '*engineering discipline'* are replaced by the words '*project management*' many project managers would consider the definition to apply to project management. This definition may be understood as applying to both the role of the systems engineer and the role of the project manager since the roles overlap both in space and time as discussed above. As such, an alternative definition is needed.

Further research to determine what went into the systems engineering stream showed that the approved Standards used in systems engineering do not seem to actually apply to systems engineering – they cover systems engineering management and the processes for engineering a system! Thus:

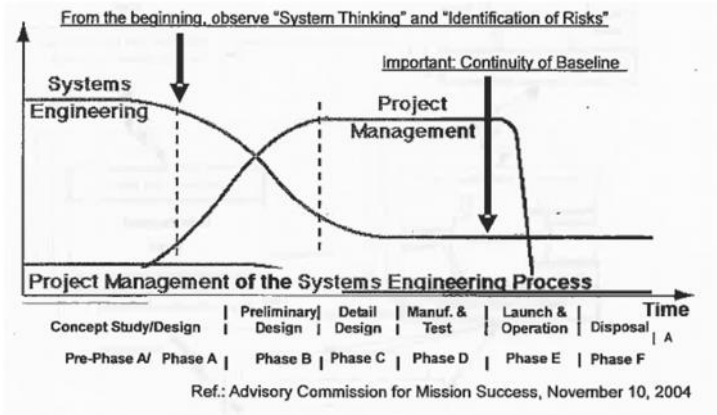


Figure 23-1 JAXA Project management and systems engineering (JAXA, 2007)

- Mil-STD-499 covers systems engineering management (MIL-STD-499, 1969).
- Mil-STD-499A covers engineering management (MIL-STD-499A, 1974) dropping the word 'systems' from the title.
- The draft (MIL-STD-499B, 1993) and MIL-STD-499C (Pennell and Knight, 2005) Standards contain the words "systems engineering" in their titles but the Standards were never approved and these Standards also (as did

499 and 499A) generally ignore most of the problem identification, whole solution conceptualisation and solution implementation planning activities that take place in the early stages of systems engineering performed in Phase A of Figure 23-1.

- ANSI/EIA-632 covers processes for engineering a system (ANSI/EIA-632, 1999).
- IEEE 1220 is for the application and management of the systems engineering process (IEEE 1220, 1998).
- TISO/IEC 15288 lists processes performed by systems engineers (Arnold, 2002) and hence may be considered as being applicable to the role of the systems engineer rather than to the activities known as systems engineering.

The phases in providing a whole complete solution to a problem can be considered as a set of activities performed by various people in various disciplines at various times. Some of those activities are systems engineering, and some are not systems engineering. The next approach was to develop a list of activities that could be described as systems engineering. Research found several sources of lists of activities including:

- Eisner lists a general set of 28 tasks and activities that were normally performed within the overall context of large-scale systems engineering (Eisner, 1988). He calls the range of activities 'specialty skills' because some people spend their careers working in these specialties. Thus according to Eisner [the role of[132]] systems engineering overlaps at least 28 engineering specialties.
- Hyer provides a list of nine activities for systems integration but which do not necessarily take place during the systems integration phase (Hyer, 1997).
- Eisner expanded his earlier list (Eisner, 1988) and discusses 30 tasks that form the central core of systems engineering (Eisner, 1997) page 156). The whole area of systems engineering management is covered in just one of the tasks. Eisner states that "*not only must a Chief Systems Engineer understand all 30 tasks; he or she must also understand the relationships between them, which is an enormously challenging undertaking that requires both a broad and deep commitment to this discipline as well as the supporting knowledge base*".

Should the research continue in this direction, the resulting list would be long, subjective and open to never ending discussion. Looking outside the box, lessons learned from psychology indicate that long lists are not the way to proceed. At one point of time in the development of theories of motiva-

[132] Author's interpretation.

tion, Henry A. Murray identified separate kinds of behaviour and developed an exhaustive list of psychogenic or social needs (Murray, 1938). However, the list is so long that there is almost a separate need for each kind of behaviour that people demonstrate (Hall and Lindzey, 1957). While Murray's list of 39 kinds of behaviours has been very influential in the field of psychology, it has not been applied directly to the study of motivation in organizations because the length of the list makes it impractical to use. On the other hand, Maslow's hierarchical classification of needs (Maslow, 1966; 1968; 1970) has been by far the most widely used classification system in the study of motivation in organizations. Maslow differs from Murray in two important ways; his list is:

- **Arranged in a hierarchy** -commonly drawn as a pyramid, and contains a set of hypotheses about the satisfaction of these needs.
- **Short** -- Only five categories.

The eventual approach chosen to determine what is and what is not a systems engineering activity was to dissolve the problem by developing a criterion for what constitutes an activity to be defined as systems engineering rather than trying to resolve the problem by a developing a list of activities. The following criterion was used to determine if an activity does or does not belong in the set of activities to be known as systems engineering:

- If the activity *deals with parts and their interactions as a whole*, then it is an activity within the set of activities to be known as systems engineering.
- If the activity deals with a part in isolation, then the activity is not an activity within the set of activities to be known as systems engineering but is part of 'something else', e.g., engineering management, software engineering, etc.

The activities of systems engineering have focused on both analysis and systems thinking. Analysis which has three steps (Ackoff, 1991) can be performed as 'reductionism' or 'decomposition' – reducing the parts to ever decreasing components in isolation, but should be performed by the systems engineer as 'elaboration' (Hitchins, 2003) pages 93-95) – which examines the parts in increasing detail *without losing track of the part's relationship to the overall system*. Systems thinking, on the other hand also has three steps (Ackoff, 1991) but they are slightly different. Comparing analysis and systems thinking in the manner shown in Table 23-1, one can see that the focus of analysis is to look inwards while the focus of systems thinking is to look outwards. Both analysis (in the form of 'elaboration') and systems thinking have their place in the activities performed in developing an understanding of a system (Hitchins, 1992) page 14) and are but two of the systems thinking perspectives (Kasser and Mackley, 2008).

Table 23-1 Analysis and systems thinking

Analysis (Machine Age)	Systems Thinking (Systems Age)
1. Take apart the thing to be understood	1. A thing to be understood is conceptualized as a part of one or more larger wholes, not as a whole to be taken apart;
2. Try to understand how these parts worked	2. An understanding of the larger system is sought;
3. Assemble an understanding of the parts into an understanding of the whole.	3. The system to be understood is explained in terms of its role or function in the containing system.

Since the activities forming the 'something else's' are part of the context of systems engineering and are often performed by systems engineers, it is recognised that systems engineers need the knowledge to perform or understand many of the activities defined as 'something else' but that knowledge *per sé* is out of the scope of the SEBoK and will be identified accordingly. The 'something else' activities were further partitioned into the following sets of non-systems engineering activities:

- Engineering.
- Management.
- Other.

The proposed activity paradigm definitions of the systems engineering sets of activities and the non-systems engineering sets of activities are as follows:

- **Systems engineering** is the set of activities involved with *dealing with parts and their interactions as a whole.*
- **Engineering** is the set of activities *dealing with a part in isolation*. If the part is not a technological product, for example if the part is such as a human element, then use of language is such that the activity is not called engineering but something else, such as training or exercising.
- **Management** is the set of activities known as planning organising, directing, staffing and controlling activities for and in the production of the *part in isolation*.
- **Other** is the remaining set of activities not included in the previous definitions.

Combining these definitions it can be seen that in the activity paradigm:

- **Systems engineering management** is the set of activities known as planning organising, directing, staffing and controlling *systems engineering* activities *in isolation from the other sets of management activities*.
- **Engineering management** is the set of activities known as planning organising, directing, staffing and controlling *engineering* activities *in isolation from the other sets of management activities.*

Lastly for the sake of completing the set of definitions, a **task** is an activity performed within a specific period of time and a **project** consists of a temporary endeavour [set of tasks] undertaken to create a unique product, service or result (PMI, 2004). It follows that:

- **Project management** is the set of activities known as planning organising, directing, staffing and controlling a temporary set of tasks undertaken to create a unique product, service or result, *in isolation from the other projects*.

The next phase in determining the SEBoK will be to identify the activities performed in each phase of the system lifecycle developing a concept of operations of the work being performed and then using the simple activity-based criterion to determine which of the activities are and which are not systems engineering. Each activity will be defined in such a manner as to terminate with the production of a tangible product or products which is/are transferred to the start of the subsequent activity (Chapter 4). The activities have been grouped by the phases in the first and second systems engineering processes[133] in the system lifecycle for a system that is developed from conception to disposal[134] using the HKMF shown in Figure 21-3. Each area of the HKMF can potentially contain all sets of (systems engineering and non-systems engineering) activities – some more than others. Figure 23-1 also provides an indication of the relative ratios between the sets of activities known as systems engineering and the sets of activities known as project management over the SLC.

Use of the HKMF has also identified one reason for debates in the meaning of terminology used by systems engineers. Words such as 'capability' and 'system design' have different meanings in different areas of the HKMF. The confusion in the use of the term 'operations concept' and 'concept of

[133] The first systems engineering process deals with identifying the real problem and a number of alternative conceptual solutions followed by the choice of an optimal conceptual solution to the whole problem, The second systems engineering process follows the first and deals with the creation, operation and disposal of an optimal physical implementation of the conceptual solution to the problem generated by the first systems engineering process.

[134] Other lifecycles do exist.

operations' can be similarly be clarified when one realizes that the terms refer to products produced in different columns of the HKMF. In addition the vocabulary for describing concepts in Layer 2 for single system development in isolation is different to the vocabulary used in Layer 3 to express the same concepts in business processing reengineering.

This approach to determining the contents of the SEBoK is also domain independent but recognises that systems engineers do need domain knowledge (as well as systems thinking, communications and interpersonal skills). A serendipitous outcome of this approach which needs more research, would truly reengineer the work of Taylor (Taylor, 1911). For example,

- The potential exists to redraw role boundaries to align with the activity boundaries and remove much of the role overlap and inefficiency in organisations.
- A systems engineering approach can be used to determine the systems and non-systems engineering activities performed in any row and column of the HKMF based on the operations performed in that area of the framework. The activities can be grouped in various ways into specific roles (job positions) and the knowledge requirements for those roles can be developed. These requirements would provide the knowledge component requirement for the person or persons to be assigned to perform the activities. The competency requirement for the person would be determined separately.

23.3 The five types of systems engineers

The human side of systems engineering is the systems engineers who perform the roles known as systems engineering. These roles perform the conceiving and creating the solution system systems engineering activities, the project management activities, engineering and other speciality engineering activities in various mixes depending on the phase in the system lifecycle and the organisation in which the systems engineer works. Optimal performance of each of the activities requires different characteristics in the systems engineer. Previous attempts to identify characteristics of systems engineers have been based on the traits attributable to systems engineers e.g. Hall, Frank and the INCOSE UK Systems Engineering Competencies Framework (Hall, 1962; Frank, 2006; Hudson, 2006) The list of desirable traits is increasing steadily. However, the lessons learned from psychology discussed above suggest lists are not the way to proceed and that an alternate approach be found. Hence, instead of using lists of traits, an alternative approach[135] of

[135] Based on years of observations by the authors.

characterising systems engineers into the following five types is proposed based on their ability to deal with problems and solutions.

- **Type I**. This type is an apprentice who has to be told "how" to implement the solution system.
- **Type II**. Type IIs are imitator/doers. This type is the most common type of systems engineer. Type IIs have the ability to follow a process to implement a physical solution system once told what to do.
- **Type III**. Type IIIs are problem solvers. Once given a statement of the problem, this type has the expertise to conceptualize the solution system and to plan the implementation of the solution, namely create the process to realize the solution.
- **Type IV.** Type IVs are problem formulators. This type has the ability to examine the situation and define the problem (Wymore, 1993) page 2), but cannot conceptualise a solution.
- **Type V.** This type is rare and combines the abilities of the Types III and IV, namely has the ability to examine the situation, define the problem, conceptualise the solution system and plan and manage the implementation of the physical solution.

A person grows through the types with education and experience. It is important to identify people with the potential to become Type Vs as early as possible in their careers[136] and then to provide them with fast track training to enable their organization to obtain the best use of their capabilities in the future. Categorization by type is also situational because a Type V when moving to a different domain can drop down to a lower level, and then, as they learn more about the domain, rise back to Type V.

23.4 A benchmark of systems engineering postgraduate degree syllabi

A benchmark of systems engineering postgraduate degree syllabi seems to indicate that:

- Much of systems engineering is now taught as declarative and procedural knowledge (Woolfolk, 1998) shown in Table 23-2 describing the second systems engineering process. To be fair, this is not unique to systems engineering (Microsoft, 2008b). And, as another example, Peter Drucker wrote *"Throughout management science - in the literature as well as in the work in progress--the emphasis is on techniques rather than principles, on mechanics rather than decisions, on tools rather than on results, and, above all, on efficiency of the part rather than on per-*

[136] These are the potential future leaders.

formance of the whole" (Drucker, 1973) page 509.) Today's academic institutions seem to be producing Type II systems engineers and managers (engineer leaders); but they should be producing or at least identifying personnel with Type V characteristics by teaching conditional knowledge.

- Some academic institutions teaching systems engineering are leaving out the critical first systems engineering process of HKMF Column A. For example, a proposed reference curriculum for systems engineering (Jain and Verma, 2007) begins in Column B of the HKMF. This reference curriculum complies with the content of Martin, Eisner, Wasson and DOD 5000 (Martin, 1997) page 95), (Eisner, 1997) page 9), (Wasson, 2006) page 60) and (DOD 5000.2-R, 2002), pages 83-84) which consider requirements as one input to the systems engineering process as mentioned above[137]. This failure to teach the critical first systems engineering process has resulted in (1) at least one generation of "systems engineers" who are unfamiliar with the critical activities in Column A of the HKMF and (2) the terms CONOPS and 'operations concept' being used interchangeably by some systems engineers who do not have an appropriate frame of reference to understand the difference between the two documents when old timers try to explain it to them.

Table 23-2 Types of knowledge (Woolfolk, 1998)

Declarative knowledge	Knowledge that can be declared in some manner. It is "knowing that" something is the case. Describing a process is declarative knowledge.
Procedural knowledge	Knowing how to do something. It must be demonstrated; performing the process demonstrates procedural knowledge.
Conditional knowledge	Knowing when and why to apply the declarative and procedural knowledge.

23.5 Hypothesis for a reason for the failure of systems engineering

The new approach to characterizing systems engineers provides a hypothesis for a reason for the failure of systems engineering in the early stages of large projects (Hiremath, 2008) and other examples of poor systems engineering implementation (GAO, 2006). For example, the cost and schedule overruns in the JSF development project shown in Table 23-3 were predicted in Sec-

[137] See Chapter 28.

tion 11.10 and hence probably preventable. Had Type V systems engineers been working on the phases of the JSF project in column A of the HKMF, the factors identified as potential causes of cost and schedule overruns leading to the prediction in Section 11.10 would have probably been identified as risks. Appropriate risk management techniques would then have been recommended and if these risk management techniques had been implemented[138], the ensuring cost and schedule overruns would have been reduced.

Table 23-3 Failure data from GAO report 06-368, 2006

Cost and Schedule Outcomes Sorted by Percent of Product Development Remaining			
Programs	Percent cost growth*	Schedule growth, in months	Percent of development remaining
Aerial Common Sensor	45%	24	85%
Future Combat System	48%	48	78%
Joint Strike Fighter	30%	23	60%
Expeditionary Fighting Vehicle	61%	48	49%
C-130 Avionics Modernization Program	122%	Delays anticipated	Undetermined
Global Hawk (RQ-4B)	166%	Delays anticipated	Undetermined

Sources: DOD (data); GAO (analysis and presentation).

*Cost growth is expressed as the percent change in program development cost estimates in 2005 base year dollars.

Research seems to show that the early systems engineers of the 1950's and 1960's tended to focus on identifying the problem (Wymore, 1993) and finding an optimal solution (Hall, 1962; Goode and Machol, 1959). These early systems engineers were of Type III, IV, and V, while the systems engineers who came later tended to focus on processes (Type II)'s. Back in the "good old days" of systems engineering Type III, IV and V systems engineers solved/resolved/dissolved the problem in the first 'systems engineering' process addressing the conceptual solution, then initiated the implementation of the solution, and moved on to the next contract, leaving the Type II's to continue assisting the development of the solution system in the second systems engineering process. There then came a time when there was a lack of new projects and so many of the Type III, IV and V's were laid off and lost to the discipline. When the need for systems engineers picked up again, in general only the Type II systems engineers were left and they took over systems engineering. They had seen a successful process for developing systems and so their focus was on the second systems engineering process. They wrote the standards used in systems engineering (MIL-STD-499, 1969; MIL-STD-499A, 1974; EIA 632, 1994; IEEE 1220, 1998) for other Type II systems engineers to follow. These Standards in turn became the foundation

[138] A big "if" since political considerations in the Type II process paradigm would probably have precluded the risk mitigation activities.

for educating systems engineers. The 499, 499A, 632, 1220, and 15288 Standards cover the systems engineering process and engineering management because there is actually very little systems engineering (the activity not the role) in the subsystem design, construction, and unit testing phases (HKMF Columns C, D and E) of the systems lifecycle for a single system in isolation. Activities pertaining to subsystems and units in isolation are engineering of systems not systems engineering activities according to the criterion defined above. The mantra became 'follow the process and all will be well'. The term GIGO - garbage in, garbage out, was acknowledged but ignored. In this paradigm:

- While the process camp subset of the systems engineering profession focuses on processes (Type II systems engineers), the literature on "excellence" focuses on people (Type V systems engineers) (Rodgers, et al., 1993; Peters and Waterman, 1982; Peters and Austin, 1985).
- The focus is on process and not on providing an understanding of the context and the ability to tailor the process as was called out in (MIL-STD-499, 1969). This is seen in systems engineering courses where the students are taught about the process but not about the context.
- Processes seen to work in one culture or organization have been copied verbatim by other organizations, with dismal results. Examples can typically be found in the lessons learned (O'Toole, 2004) and some reasons for a claimed Six Sigma initiative 60% failure rate (Angel and Froelich, 2008).
- The systems engineering process has a high degree of correlation to the problem solving process because that was the process documented in the Standards (Section 25.3).
- The Standards commonly used/taught in systems engineering (MIL-STD-499, 1969; MIL-STD-499A, 1974) and (DOD 5000.2-R, 2002) pages 83-84) ignore most of the activities allocated to Phase A in Figure 23-1 and Phase A of the HKMF resulting in the critical first systems engineering process addressing the conceptual solution being out of mainstream Type II systems engineering. Table 23-4 contains data extracted from Table 5 in (Honour and Valerdi, 2006) and rearranged in chronological order[139] showing the lack of coverage of the mission purpose/definition activities in MIL-STD 499 and ANSI EIA 632. The top row in Table 23-4 has been added in this Chapter to show that MIL-STD 499 and ANSI EIA 632 do not cover the conceptual activities in early stage systems engineering or the first systems engineering process and while the Systems

[139] Based on the issue date of MIL-STD-499, not the draft MIL-STD-499C since the contents of MIL-STD 499A and MIL-STD-499B don't differ from MIL-STD 499C in this respect.

Engineering CMM, the draft MIL-STD-499C Standard and ISO 15288 do address the mission/purpose definition activities to some extent they also do not cover the conceptual activities in the first systems engineering process. This situation (addressing mission/purpose definition activities to some extent while failing to cover the first systems engineering process) also appeared in a survey of current systems engineering processes (Bruno and Mar, 1997) and in the list of the engineering and systems engineering activities assigned to the systems engineering organization/team based on the (MIL-STD-499B, 1993)/(EIA 632, 1994) Standards (Fisher, 1996).

Table 23-4 Focus of Standards – chronological order

SE Categories	MIL-STD-499C	ANSI/ EIA 632	IEEE-1220	CMMI	ISO-15288
Conceptualizing problem and alternative solutions	No	No	No	No	No
Mission/purpose definition	No	No	✓	✓	✓
Requirements engineering	✓	✓	✓	✓	✓
System architecting	No	✓	✓	✓	✓
System implementation	✓	✓	No	✓	✓
Technical analysis	✓	✓	✓	✓	✓
Technical management/ leadership	✓	✓	✓	✓	✓
Verification & validation	✓	✓	✓	✓	✓

Based on a combination of the five types of systems engineers and the history of systems engineering paraphrased in terms of those five types, the hypothesis is that a current cause of failures in systems engineering is the assignment of Type II systems engineers or higher types trained in a Type II process thinking paradigm to tasks that need the problem/solution characteristics of the Type III, IV and V systems engineers. The associated prediction to test the hypothesis is that the cost and schedule overruns and other failures will continue in spite of all the funding being allocated to systems engineering education if the education of engineer leaders remains in the Type II paradigm and starts with the activities in column B of the HKMF. Type II systems engineers are and should be doing the engineering of systems (following the process designed by the Type V systems engineers). Type V systems engineers should be doing systems engineering in Columns A, B, F and G of the HKMF.

23.6 Recommendations

The following recommendations are made to improve systems engineering based on the research so far:

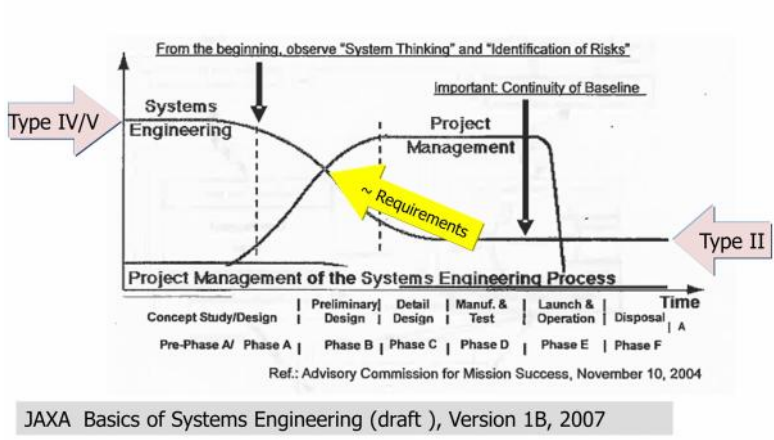


Figure 23-2 Mapping Types into SDLC

1. Continue with the development of the SEBoK creating a concept of operations for the product producing activities in each rectangle of the HKMF using the simple activity-based criterion to determine which of the activities are and which are not systems engineering and then defining the knowledge requirements for the activities known as systems engineering.
2. Investigate the potential of redrawing role boundaries to align with the activity boundaries and remove much of the role overlap and inefficiency in organisations. This approach, which needs more research, would truly reengineer the work of F. W. Taylor (Taylor, 1911).
3. Once the activities performed by systems engineers in each area of the HKMF have been identified, an appropriate level of competence for the activity should be made and optimal systems engineering teams could then be designed.
4. Work with psychologists to identify characteristics of the five types of engineer leaders so that the Type V's may be identified early in their career and put through fast track training to increase their value to their organizations.
5. Modify the curriculum for teaching systems engineering to include activities enabling the early identification of potential Type V's.
6. Modify the curriculum for teaching systems engineering to include the system engineering activities performed in Column A of the HKMF.
7. Develop a good set of educational materials for use with the modified curriculum.
8. Identify the activities performed in the non-systems engineering streams in each column of the HKMF. Then determine the

knowledge and type of engineer leader needed to make optimal decisions and quantify the risks associated with decision making with specific levels of imperfect knowledge and using the wrong type of engineer leader. This model should inform customers concerning the prediction of the probability of future project failure at any point in any column of the HKMF by comparing the situation in a real project with the data in the model.

23.7 Summary

This Chapter documented the early phases of using systems engineering to develop a SEBoK and some of the findings. The first part of the Chapter discussed the nature of the problem and dissolved the problem by applying an out-of-the-box approach. The second part of the Chapter identified five types of systems engineers, discussed the evolution of systems engineering in terms of those five types, hypothesised that a major cause of the failure of systems engineering is the allocation of inappropriate types of systems engineers to systems engineering activities and identified a critical gap in systems engineering education. The Chapter concluded with some insights from the out-of-the-box approach and recommendations for further study.

23.8 Conclusion

The out-of the-box approach to developing the SEBoK seems to be achievable and has produced some interesting insights.

2013

24 A framework for benchmarking competency assessment models

This Chapter continues from where section 7.2 ended and discusses the need for competent systems engineers, the differences between nine current ways of assessing competencies (competency models) and the difficulty of comparing the competency models due to the different ways each model groups the competencies. The Chapter then introduces a competency model maturity framework (CMMF) for benchmarking competency models of systems engineers. The Chapter benchmarks the nine models using the CMMF and a surprising finding was an error of omission in all nine models. The Chapter shows that the CMMF can also be used as the basis for developing an original model for a specific organisation in a specific time and place and concludes with suggestions for future research.

24.1 Introduction

Current approaches for constructing and using competency models are based on observations of what systems engineers do in organisations but there is no way to directly compare competency models or verify if indeed they are fit for purpose. The purpose of this Chapter is to introduce a CMMF for comparing or benchmarking competency models of systems engineers. The literature review for the research documented in this Chapter covers both systems engineering and the domain of cognitive psychology. The Chapter begins with a discussion on the need for competent systems engineers, then discusses the role of systems engineers in the workplace providing examples of a number of points of view as to what those roles are and alludes to the difficulty of gaining a generic consensus on the nature of the role of the systems engineer. At that point the focus of the Chapter changes to ways of assessing the competencies of systems engineers. The Chapter continues with a brief examination of nine different competency models and shows that each competency model seems to have been designed for a different purpose in different domains, times and places. Since competency and competency models are widely discussed in the domain of cognitive psychology, the authors have found it both necessary and helpful to adopt

language from the domain of cognitive psychology, avoiding the need to invent yet more systems engineering terminology for concepts already well-defined in the domain of cognitive psychology. Table 24-1 contains a glossary to assist in the comprehension of the Chapter.

The Chapter then introduces a two-dimensional CMMF that can be used for comparing and enhancing existing competency models and as a competency model by those organisations that do not yet have a competency model and wish to develop one. The CMMF is based on assessing the competency needed to perform systems engineering in five monotonically ascending levels. The Chapter then benchmarks the nine competency models studied using the CMMF, identifies an error of omission common to all models, and continues with suggestions for future research.

Table 24-1 Glossary

Word	Meaning	Source
Ability	The required competence to perform the function successfully.	
Analytical thinking	The abstract separation of a whole into its constituent parts in order to study the parts and their relations	TheFreedictionary.com
Behaviour	The way in which an animal or a person acts in response to a particular situation stimulus	Oxford American Dictionary
Capability	Having the ability required for a specific task or accomplishment	TheFreedictionary.com
Cognitive	of, relating to, being, or involving conscious intellectual activity (as thinking, reasoning, or remembering)	Merriam-Webster, 2011
Competencies	Behaviours that encompass the knowledge, skills, and attributes required for successful performance	(LaRocca, 1999)
Competency model	A descriptive tool that identifies the competencies needed to operate in a specific role within a(n) job, occupation, organization, or industry.	(Ennis, 2008)
	A collection of competencies that together define successful performance in a particular work setting.	(ETA, 2010)
Competent	having requisite or adequate ability or qualities	Merriam-Webster, 2011

Conditional knowledge	Knowing when and why to apply the declarative and procedural knowledge.	(Woolfolk, 1998)
Critical thinking	Judicious reasoning about what to believe and therefore what to do	(Tittle, 2011)
Declarative knowledge	Knowledge that can be declared in some manner. It is "knowing that" something is the case. Describing a process is declarative knowledge.	(Woolfolk, 1998)
Holistic thinking	Art and science of handling interdependent sets of variables	(Gharajedaghi, 1999)
	Thinking about a system as a whole but also doing the thinking in a systemic and systematic manner.	(Kasser, 2013) Chapter 6
Knowledge	A body of information needed for the successful performance of the process, or set of processes, considered relevant to the discipline of interest.	
Procedural knowledge	Knowing how to do something. It must be demonstrated; performing a process demonstrates procedural knowledge.	(Woolfolk, 1998)
Proficiency	Levels of capability	(Metzger and Bender, 2007)
Skill	The observable or measured competence in performing the function.	
Systems thinking	Systems thinking seeks to address and solve complex problems by understanding the system parts and their interactions within the context of the whole system, rather than in isolation. C.f. Systems Approach	Hitchins, 2011
Trait	a distinguishing quality (as of personal character)	Merriam-Webster, 2011

24.2 The need for competent systems engineers

Inadequate systems engineering is repeatedly cited as a major contributor to failed projects particularly in NASA and the US DOD (Evans, 1989; Leveson, 2004; Welby, 2010; Wynne and Schaeffer, 2005). A literature review reveals that many of the works on improving systems engineering have focused on improving and developing new systems engineering processes, and tend to ignore people (Swarz and DeRosa, 2006; Goldberg and Assaraf, 2010). To be fair, this is not unique to systems engineering (Microsoft, 2008a). For exam-

ple, Peter Drucker wrote *"Throughout management science--in the literature as well as in the work in progress--the emphasis is on techniques rather than principles, on mechanics rather than decisions, on tools rather than on results, and, above all, on efficiency of the part rather than on performance of the whole"* (Drucker, 1973) page 509.

However, the literature on "excellence" focuses on people and ignores process (Peters and Waterman, 1982; Peters and Austin, 1985; Rodgers, et al., 1993). In addition, Robert A. Frosch, when an assistant secretary to the US Navy, wrote, "*systems, even very large systems, are not developed by the tools of systems engineering, but only by the engineers using the tools*" (Frosch, 1969). And out of the software realm comes the phrase attributed to Grady Booch "*a fool with a tool is still a fool*".

While the focus on improving process continues (Goldberg and Assaraf, 2010; West, 2010), the need to certify the competencies of systems engineers is now becoming widely recognised. For example, systems engineering competency models are becoming more widely developed and used in support of systems engineering workforce selection, development, education and training (Burke, et al., 2000; Jansma and Jones, 2006; Verma, et al., 2008; Menrad and Larson, 2008; Squires, et al., 2011). In addition there is a growing international interest in a certified systems engineering professional (CSEP) qualification[140].

24.3 Roles and activities of systems engineers

Research into developing the requirements for, and subsequently updating, a 21st century introductory immersion course on systems engineering (Kasser, et al., 2008), included a literature review of text books published between 1959 and 2009 starting with Goode and Machol (Goode and Machol, 1959) as well as the proceedings of the international symposia of the INCOSE since 1991. Findings from this research determined that:

- The role of the systems engineer in the workplace depends on the situation.
- Definitions and descriptions of systems engineering currently comprise different interpretations of the broad raft of activities that systems engineers might undertake according to their role in the workplace.

This multichotomy exists because different users of the term 'systems engineering' for almost 60 years have chosen or perceived different meanings. For example, one comment from 1960 was "*Despite the difficulties of*

[140] There are independent national qualifications in Korea and Singapore.

finding a universally accepted definition of systems engineering[141]*, it is fair to say that the systems engineer is the man who is generally responsible for the over-all planning, design, testing, and production of today's automatic and semi-automatic systems*" (Chapanis, 1960) page 357). Jenkins expanded that comment into 12 roles of a systems engineer (Section 23.1) and seven of those roles (activities performed by a person with the title systems engineer) overlapped the role of the project manager (activities performed by a person with the title project manager) (Jenkins, 1969) page 164).

Some of the evolution of the role of the systems engineer can be seen in the very little overlap between the 12 roles documented by Jenkins and Sheard's 12 systems engineering roles (Section 23.1). Jenkins' roles relate to the activities in conceiving and planning the solution system while almost 30 years later, Sheard's set of roles relate to interpersonal relationships between the practitioners of disparate skills and disciplines implementing the solution system.

Furthermore, the role of the systems engineer (the activities performed by a person with the title systems engineer) overlaps the roles of people from other professions according to both Jenkins and Sheard. The research found several other sources of lists of activities performed by systems engineers including:

- A few examples of the different overlaps between systems engineering and project management (DSMC, 1996; Brekka, et al., 1994; Roe, 1995; Sheard, 1996; Johnson, 1997; Watts and Mar, 1997; Bottomly, et al., 1998; Kasser, 1996)[142]. In addition, the Defense Systems Management College definition of systems engineering is "*The management function which controls the total system development effort for the purpose of achieving an optimum balance of all system elements. It is a process which transforms an operational need into a description of system parameters and integrates those parameters to optimise the overall system effectiveness*" (DSMC, 1996). Notice the use of the term "*management function*".
- A discussion on the overlaps between systems engineering and other disciplines (Emes, et al., 2005).
- An example of the activities performed in new product design that overlaps those of systems engineering (Hari, et al., 2004).
- A general set of 28 tasks and activities that were normally performed within the overall context of large-scale systems engineering (Eisner, 1988). Eisner called the range of activities 'specialty skills' because some people spend their careers working in these specialties. Thus ac-

[141] Fifty years later, nothing has changed in that respect.
[142] A different set of overlaps, as seen across the years.

cording to Eisner [the role of][143] the systems engineer overlaps at least 28 engineering specialties.

- 30 tasks that form the central core of systems engineering (Eisner, 1997) page 156. The whole area of systems engineering management is covered in just one of the tasks. Eisner states that "*not only must a Chief Systems Engineer understand all 30 tasks; he or she must also understand the relationships between them, which is an enormously challenging undertaking that requires both a broad and deep commitment to this discipline as well as the supporting knowledge base*".

24.4 Assessing systems engineering competency

Competency assessment tends to be performed using competency models which form the foundation for developing curriculum and selecting training materials, and for licensure and certification requirements, job descriptions, recruiting and hiring, and performance reviews (CareerOneStop, 2011). "*These models have competency domains broken down into competency groups and further sub-categorized into sub-competencies. As one continues to the next*[144] *levels in the hierarchy, the competencies become further focused and specific to the industry, job or occupation, and position*" (Ennis, 2008). A multi-level assessment approach to assessing proficiencies of systems engineers groups the knowledge, traits, abilities and other characteristics of successful systems engineers into a two-dimensional maturity model[145] in accordance with Arnold who wrote "*at its simplest, competence may be viewed in terms of two dimensions or axes. One axis defines the process, or set of processes, considered relevant to the discipline of interest. The other axis establishes the level of proficiency attained typically using a progression of increasing-value cardinal points that are defined in terms of attainment or performance criteria*" (Arnold, 2000) as shown in Figure 24-1.

The activities performed by a systems engineer in one organisation are different to those performed by a systems engineer in another organization and sometimes even in different parts of the same organisation (Section 19.7). It could thus be expected that different ways of assessing the competency of systems engineers would assess different characteristics. The following four competency models were studied to determine their coverage.

[143] Author's interpretation.
[144] Next level down, or lower levels.
[145] Due to space limitations, where prior work covers a topic in detail, the work is cited and summarized.

Figure 24-1 Two dimensional assessments

1. Knowledge, Skills, and Abilities (KSA).
2. INCOSE Certified Systems Engineer Professional (CSEP) Examination (INCOSE, 2008).
3. INCOSE UK Systems Engineering Competencies Framework (SECF) (INCOSE UK, 2010).
4. Capacity for Engineering Systems Thinking (CEST) (Frank, 2006).

The findings showed that each of the competency models had different goals and objectives (Kasser, et al., 2010). Sometime later in the research, the following competency models were also studied with similar findings.

5. A systems engineering competency taxonomy (SECT) (Squires, et al., 2011).
6. NASA 2010 Systems Engineering Competencies (NASA, 2010).
7. The JPL Systems Engineering Advancement (SEA) project (Jansma and Jones, 2006).
8. MITRE 2007 Systems Engineering Competency Model (Metzger and Bender, 2007).
9. National Defense Industrial Association (NDIA) proposed systems engineering competency model (Gelosh, 2008).

Consider each of these competency models. Descriptions of each competency model are brief and where details of the content are given, the intent of each summary is to enable the differences between the competency models to be seen, not to highlight the contents of the competency model.

24.4.1 Knowledge, Skills, and Abilities

Knowledge, Skills, and Abilities (KSA) are one way of assessing the suitability of candidates for job positions according to qualification standards published by the US Office of Personnel Management (OPM). These standards are intended to identify applicants who are likely to perform successfully on the job, and to screen out those who are unlikely to do so (OPM, 2009). In practice, KSAs tend to be lists of statements written by, or on behalf of, candidates. These statements are targeted to specific positions and describe a

number of situational challenges faced by the candidate and outcomes achieved in previous jobs that are to be used by evaluators in a pass-fail mode when looking for qualified candidates for the specific position.

KSAs are an improvement over resumes written as job descriptions citing years of experience that state nothing about the achievements of the person. Moreover, being descriptive, KSAs do not seem to be generally suitable for assessing the difference between a person who does not understand the underlying fundamentals and just follows a process to reach a successful conclusion and a person who understands what needs to be done and can create and implement a process to do it successfully. Lastly, while KSAs can provide a multi-level assessment of the proficiency of a systems engineer, there is no standard definition for any such levels.

24.4.2 INCOSE CSEP Exam

The INCOSE Certified Systems Engineering Professional (CSEP) examination (INCOSE, 2008) is only a part of the three-tier INCOSE approach to certifying the competency of a systems engineer and should not be considered as a stand-alone certification of competency. The INCOSE CSEP examination is designed to test the applicant's knowledge of the contents of the INCOSE Systems Engineering Handbook (Haskins, 2011; 2006a). Consequently, the handbook focuses on processes according to ISO/IEC 15288, only addresses a limited body of declarative and procedural knowledge and does not address the cognitive skills and the individual traits in an objective manner. These skills and traits are addressed in a subjective manner in the follow up evaluation of the career experience of the candidate. The CSEP examination may be considered as a minimal measurement of systems engineering competency.

24.4.3 INCOSE UK Systems Engineering Competencies Framework

The Systems Engineering Competency Framework (SECF) (INCOSE UK, 2010) was initially developed in response to an issue identified by the INCOSE UK Advisory Board (UKAB) (Hudson, 2006). The objective determined by the INCOSE UKAB was "*to have a measurable set of competencies for systems engineering which will achieve national recognition and will be useful to the enterprises represented by the UKAB"*. The focus of the SECF is on the competencies of systems engineering rather than the competencies of a systems engineer.

The SECF competencies are grouped into three themes; Systems Thinking, Holistic Lifecycle View, and Systems Engineering Management.

1. **Systems Thinking** contains the underpinning systems concepts and the system/super-system skills including the enterprise and technology environment.

2. **Holistic Lifecycle View** contains all the skills associated the systems lifecycle from need identification, requirements through to operation and ultimately disposal.
3. **Systems Engineering Management** deals with the skills of choosing the appropriate lifecycle and the planning, monitoring and control of the systems engineering process.

According to the SECF, each competency should be assessed in terms of four levels of comprehension and experience defined by "Awareness" through to "Expert".

1. **Awareness:** The person is able to understand the key issues and their implications. They are able to ask relevant and constructive questions on the subject. This level is aimed at enterprise roles that interface with Systems Engineering and therefore require an understanding of the Systems Engineering role within the enterprise.
2. **Supervised Practitioner:** The person displays an understanding of the subject but requires guidance and supervision. This proficiency level defines those engineers who are "in-training" or are inexperienced in that particular competency.
3. **Practitioner:** The person displays detailed knowledge of the subject and is capable of providing guidance and advice to others.
4. **Expert:** The person displays extensive and substantial practical experience and applied knowledge of the subject.

While the SECF is a worthwhile effort, there seem to be a number of inconsistencies in the document including:

- The four levels of proficiency are not in the same dimension: while the last three levels are attributable to increasing levels of proficiency of systems engineers, the 'awareness' level is applicable to people who work with systems engineers at high levels in an organization and as such there is an assumption that these people should have some knowledge of systems engineering.
- The allocation of knowledge to the systems thinking competency theme does not match the way the term cognitive skills is used in the systems thinking and critical thinking professions (domains). This is a potential cause of confusion.
- While lists of abilities within the competencies make it easy to assess compliance by checking off experience against the items on the list, the method has the same intrinsic defect as the use of KSAs. Namely, it does not seem to be generally suitable for assessing the difference between a person who does not understand the underlying fundamentals and just follows a process to reach a successful conclusion and a person who understands what needs to be done and can create and implement a process to do it successfully.

The SECF does however provide a way of setting the systems engineering role proficiency requirements for jobs in a process-oriented work environment, namely meets the one of the purposes for competency models produced by human resource professionals. Nevertheless, it should be used with care for assessing the competencies of individuals due to:

- its lack of an objective way of assessing cognitive skills and individual traits;
- its being based on the observed role of a systems engineer in a number of UK organisations; namely the knowledge that systems engineers in the UK have, rather than the knowledge systems engineers need to have.

24.4.4 Capacity for Engineering Systems Thinking (CEST)

The capacity for engineering system thinking (CEST) is a proposed set of high order thinking skills that enable individuals to successfully perform systems engineering tasks (Frank, 2006). A study aimed at identifying the characteristics of successful systems engineers identified 83, which were aggregated into four sets of characteristics as follows:

1. cognitive characteristics related to systems thinking,
2. systems engineering skills,
3. individual traits,
4. multidisciplinary knowledge and experience.

CEST focuses on the cognitive skills, individual traits, capabilities and knowledge and background characteristics of a systems engineer who can examine system failures and identify and remedy system problems (Frank and Waks, 2001). As such, it may be useful for assessing these aspects of the competency of systems engineers. However, at this time, CEST is still in its research stages.

24.4.5 A systems engineering competency taxonomy (SECT)

Squires, Wade, Dominick and Gelosh have built a systems engineering competency taxonomy (SECT) from a selected set of existing competency models combined with some systems thinking (Squires, et al., 2011).

The authors combined the following three models into single Experience Accelerator (ExpAcc) competency taxonomy:

1. The Systems Planning, Research, Development, and Engineering (SPRDE) Systems Engineering and Program Systems Engineer (PSE) competency model, known as the SPRDE-E/PSE (DAU, 2010);
2. The Systems Engineering Research Center (SERC) Technical Lead Competency Model (Gavito, et al., 2010);

3. A Critical/Systems Thinking Competency Model (Squires, 2007).

The final SECT competency taxonomy which covers 87 unique competencies is based on the following three-pronged approach:

- Systems and critical thinking is the backbone of the model.
- Technical expertise which comprises technical leadership, technical management, and technical/analytical skills.
- Project management and other broad-based professional competencies.

Unlike the other competency models studied, SECT evaluates the ability to deal with complexity in several levels of proficiency.

24.4.6 NASA 2010 Systems Engineering Competencies

NASA identified 49 systems engineering competencies which are grouped by competency areas, competencies and competency elements and assessed in four proficiency levels (NASA, 2010).

- The ten competency areas are concepts and architecture, system design, production and operations, technical management, project management, internal and external environments, human capital management, security and safety, professional and leadership development.
- The 35 systems engineering element competencies express the overall knowledge, skills and behaviours that systems engineers are expected to possess and/or perform as a part of their job.
- The four proficiency levels are technical engineer/project team member, subsystem lead, project systems engineer and program systems engineer.

The model is tailored to NASA's needs. It does not include any overt reference to systems thinking, cognitive competencies and behavioural traits.

24.4.7 The JPL Systems Engineering Advancement (SEA) project

Jansma and Jones developed a systems engineering competency model along three axes; processes, personal behaviours and technical knowledge as part of a project to improve systems engineering at the Jet Propulsion Laboratory (JPL) (Jansma and Jones, 2006). The SEA Project utilized a rigorous process to identify a list of highly valued personal behaviours of systems engineers.

- **The processes axis** encompasses ten systems engineering functions.
- **The personal behaviours** fall into five clusters –

1. leadership skills,
2. attitudes and attributes,
3. communication,

4. problem solving and systems thinking, and
5. technical acumen.

- **The technical knowledge** axis encompasses 21 systems engineering disciplines and fields.

24.4.8 MITRE 2007 Systems Engineering Competency Model

The MITRE Systems Engineering competency model (Metzger and Bender, 2007) is based on criteria for successful MITRE systems engineers. The MITRE model has three cumulative levels of proficiency (i.e., levels of proficiency) and consists of 36 competencies organized into five sections:

1. Enterprise Perspectives.
2. Systems Engineering Lifecycle.
3. Systems Engineering Planning and Management.
4. Systems Engineering Technical Specialties.
5. Collaboration and Individual Characteristics.

The authors of this model do not claim that their model is a general competency model. They explicitly state that the model is tailored to the MITRE needs. The model was not "scientifically" validated. The authors generally claim that "*the original draft competencies were based upon information from standards bodies, the MITRE Institute, commercial companies, and Government sources … The model went through numerous revisions with input from many people across MITRE before it reached this form. It will continue to evolve and be upgraded …*"

The MITRE systems engineering competency model has three increasing levels of proficiency, Foundational, Intermediate, and Expert. MITRE assumes that a person's competence at a specific proficiency level is generally the result of education, work experience, job tasks, and specific job roles. A MITRE systems engineer is likely to be expert in some competencies, intermediate in others, and foundational in others. It is not expected, and it would be highly unlikely, that any one person would be expert in all the behaviours and competencies in this model.

24.4.9 National Defense Industrial Association (NDIA) proposed systems engineering competency model

The National Defense Industrial Association (NDIA) proposed systems engineering competency model groups 50 competencies in the following four areas (Gelosh, 2008):

1. **Analytical** containing 20 competencies covering systems engineering tools and techniques design considerations.

Table 24-2 Arrangement of competencies in the nine competency models

KSAs	INCOSE CSEP Exam	SECF	CEST	SECT	NASA 2010	JPL SEA	MITRE	NDIA
N/A	Systems Engineering Overview	Systems Thinking	Cognitive Characteristics	Systems and Critical Thinking	Concepts and Architecture	Processes	Enterprise Perspectives	Analytical
	General Lifecycle Stages	Holistic Lifecycle View	Systems Engineering Skills	Technical Expertise	System Design	Personal Behaviors	Systems Engineering Lifecycle	Technical Management
	Technical Processes	Systems Engineering Management	Individual Traits	Project Management	Production and Operations	Technical Knowledge	Systems Engineering Planning and Management	General
	Project Processes		Multidisciplinary Knowledge		Technical Management		Systems Engineering Technical Specialties	Professional Competencies

1. **Technical management** containing 15 competencies in the technical management process.
2. **General** containing five competencies pertaining to a total systems view.
3. **Professional competencies** containing 10 competencies covering thinking, problem solving and inter-personal skills.

The planned approach, according to the presentation, was to develop the competencies based on the roles of systems engineers. Two years later, the model was still a work in progress (NDIA E&T, 2010), for example, the first of the proposed 2011 tasks was to survey existing, freely available systems engineering competency models for entry-level systems engineers to develop the minimum requirements for an individual to be called a systems engineer. Reasons for this lack of progress may include:

- The difficulty of defining a role-based SEBoK due to the broad range of non-systems engineering activities performed by systems engineers in their role in the workplace that require knowledge from other disciplines (Chapter 12, and 23).
- The different opinions on the nature of systems engineering[146] that preclude obtaining consensus with respect to a SEBoK for systems engineering.

And without consensus on a SEBoK, the committee cannot produce even a minimal objective traceable set of generic requirements for the competency of a systems engineer.

24.5 Comparing the different competency models

As expected, each of the competency models described above was developed to provide a solution to a different problem and contains different bodies of knowledge. This is in accordance with general industry practice for the design and use of competency models (Ennis, 2008). However, none of the competency models discussed above was presented in a format compatible with the nine-tier US Employment and Training Administration (ETA) Competency Model Clearinghouse's General Competency Model Framework (ETA, 2010). Each of these competency models identified a large number of competencies and then grouped the competencies into smaller manageable but different groups that while meeting the need of the time and place, make comparing the assessment approaches difficult as shown in the summary in Table 24-2.

[146] See discussion on the camps in Chapter 27.

At the detailed level, NDIA aggregates 'requirements management' into 'technical competencies' (Gelosh, 2008) while MITRE groups the same function into 'systems engineering lifecycle' (Metzger and Bender, 2007). SECT allocates 'requirements analysis' to 'Technical/Analytical Competencies' (Squires, et al., 2011) while MITRE incorporates the function into 'requirements engineering' which is allocated to 'systems lifecycle'. Thus, a common framework that could encompass all the assessment approaches is needed to compare the different competency models. This framework would allow owners and users of each of the competency models to benchmark their competency model against the others, perhaps identify gaps, and upgrade their approach.

Some of the competencies being assessed fall into the category of cognitive characteristics. The traditional academic approach to measuring cognitive characteristics is based on the revised Bloom's taxonomy which combines systems thinking and critical thinking (Anderson, et al., 2000). Research into the psychology domain identified an alternative approach which unlike Bloom's taxonomy, allows for the systems thinking and critical thinking skills to be assessed separately[147] (Kasser, 2010).

The levels of ability in each in each of the nine competency models studied are also different, some models only recognise one level, some models assess skill proficiencies and some assess necessary proficiencies for job positions (roles) at specific levels in the organisational hierarchy as shown in Table 24-3. Note that the SECT evaluates proficiency in dealing with complexity; a different scale with respect to the evaluation of proficiency in the other competency models.

•

[147] See section 24.6.1.2.

Table 24-3 Comparison of proficiency levels in the competency models

KSAs	INCOSE CSEP Exam	SECF	CEST	SECT	NASA 2010	JPL SEA	MITRE	NDIA
N/A	N/A	Awareness	N/A	None or Aware Only	Technical Engineer/Project Team Member	N/A	Foundational	N/A
		Supervised Practitioner		Apply with Guidance	Subsystem Lead		Intermediate	
		Practitioner		Apply	Project Systems engineer		Expert	
		Expert		Manage or Lead	Program systems engineer			
				Advance State of Art				

KSAs	INCOSE CSEP Exam	SECF	CEST	SECT	NASA 2010	JPL SEA	MITRE	NDIA
	Agreement Processes				Project Management		Collaboration and Individual Characteristics	
	Organizational Project Enabling Processes				Internal and External Environments			
	Tailoring Processes				Human Capital Management			
	Specialty Engineering Activities				Security and Safety			
					Professional and Leadership Development			

24.6 A two-dimensional competency maturity model framework for benchmarking the competency models of systems engineers

This section introduces a CMMF for benchmarking the different competency models.

24.6.1 The vertical dimension

The vertical dimension is based on three categories:

1. knowledge,
1. cognitive characteristics, and
2. individual traits

24.6.1.1 Category 1: Knowledge

Knowledge covers:

- systems engineering,
- the problem domain,
- the implementation domain in which the system is being realized,

the solution domain in which the systems engineering is being applied to realize the solution system.

Opinions vary on what constitutes systems engineering; each opinion will have a different vision of the knowledge content. This was reflected in the different ways of assessing systems engineering proficiency discussed above. In addition, since systems engineers apply their skills in different domains (e.g. aerospace, land and marine transportation, information technology, defence, etc.), there is an assumption that to work in any specific domain, the systems engineer will need the appropriate problem, solution and implementation domain knowledge.

Knowledge of "systems engineering process" and systems engineering tools is considered as part of systems engineering rather than the implementation domain. The implementation domain sets the constraints on both the process and solution system. For example, the development system for a software system is in the implementation domain. Implementation domain knowledge relates to the properties of the compiler as well as the characteristics (especially limitations) of the development hardware. In another environment, the implementation domain might include thermal vacuum chambers and other equipment used to partially or fully develop and test the solution system.

It is tempting to assume that the problem domain and the solution domain are the same, but they are not necessarily so for example, the problem domain may be urban social congestion, while the solution domain may be a

form of underground transport to relieve that congestion. Lack of problem domain competency may lead to the identification of the wrong problem and lack of solution domain competency may lead to selection of a less than optimal or even an unachievable solution system. Risk management is an activity (process) that requires competency in the problem, solution and implementation domains. Similarly, a "system failure" needs to be distinguished from a "system problem".

The large number of implementation, problem and solution domains in which systems engineering takes place also requires a corresponding large SEBoK which is not necessarily applicable to all systems engineers. Consequently, the knowledge must be tailored to the specific problem, solution and implementation domains and the phase of the system lifecycle. For example:

- **Requirements analysis.** Systems engineers performing requirements analysis will need to know how to develop a matching set of specifications that describe the mission and support functions of the solution system in its fielded operational context; a different subset of systems engineering knowledge to that needed by systems engineers performing test and evaluation.
- C**ontrol or operations and maintenance environment.** Systems engineers working in a control or operations and maintenance environment will need knowledge of the software development process and the tools, and the properties of the underlying development hardware platforms as well as the solution domain in which the system is to be fielded.
- **Electro-optical engineering.** Systems engineers working in an electro-optical engineering factory will need knowledge of how the various components can be configured without disturbing the performance of the system.
- **Socio-technical systems.** Systems engineers working on socio-technical systems will need the appropriate knowledge of human behaviour and how humans interact with technology and each other.
- And so on.

24.6.1.2 Category 2: cognitive characteristics

Cognitive characteristics namely systems thinking and critical thinking provide the problem identification and solving skills[148] to think, identify and tackle problems by solving, resolving, dissolving or absolving problems (Ackoff, 1999) page 115), in both the conceptual and physical domains.

[148] Problem solving and identification skills have been listed separately to map into Type IV and V as discussed below.

Problem identification and solving competency is not the same thing as problem domain competency.

24.6.1.2.1 holistic thinking

The approach to the assessment of systems thinking was developed from the only systematic and systemic approach to applying systems thinking discovered in the literature (Richmond, 1993). Further research[149] based on Richmond's work produced a set of nine viewpoints called System Thinking Perspectives (STP) (Kasser and Mackley, 2008) which evolved into the Holistic thinking perspectives (HTP) (Kasser, 2013) and have been used in teaching holistic thinking in postgraduate classes and workshops in Israel, Japan, Singapore, Taiwan and the UK. The HTPs are Operational, Functional, Big Picture, Structural, Generic, Continuum, Temporal, Quantitative and Scientific. Of these nine perspectives, the first eight are descriptive and the ninth is prescriptive. The eight descriptive perspectives are used to view or describe the situation, while the prescriptive perspective is the one which contains the statements of the problem and candidate solutions.

24.6.1.2.2 Critical thinking

A literature review showed that the problem of assessing the degree of critical thinking in students seemed to have already been solved (Eichhorn, 2002; Wolcott and Gray, 2003; Allen, 2004; Paul and Elder, 2006). Wolcott and Gray aggregated lists of critical thinking abilities by defining five levels of critical thinking (Wolcott and Gray, 2003). In addition, Wolcott's method for assessing a critical thinking level was very similar to that used by Biggs for assessing deep learning in the education domain (Biggs, 1999). Since a tailored version of the Biggs criteria had been used successfully at the University of South Australia for assessing student's work in postgraduate classes on systems engineering (Kasser, et al., 2005), Wolcott's method was adopted for the CMMF. Wolcott's five levels (from lowest to highest) are:

0. Confused fact finder.
1. Biased jumper.
2. Perpetual analyzer.
3. Pragmatic performer.
4. Strategic re-visioner.
5. Consider each of them.

24.6.1.2.2.1 Confused fact finder

A confused fact finder is a person who is characterised by the following:

- Looks for the "only" answer.

[149] Funded by a grant from the Leverhulme trust to Cranfield University in 2007.

- Doesn't seem to "get it".
- Quotes inappropriately from textbooks.
- Provides illogical/contradictory arguments.
- Insists professor, the textbook, or other experts provide "correct" answers even to open-ended problems.

24.6.1.2.2.2 Biased jumper

A biased jumper is a person whose opinions are not influenced by facts. This person is characterised by the following:

- Jumps to conclusions.
- Does not recognise own biases; accuses others of being biased.
- Stacks up evidence for own position; ignores contradictory evidence.
- Uses arguments for own position.
- Uses arguments against others.
- Equates unsupported personal opinion with other forms of evidence.
- Acknowledges multiple viewpoints but cannot adequately address a problem from viewpoint other than own.

24.6.1.2.2.3 Perpetual analyzer

A perpetual analyser is a person who can easily end up in "analysis paralysis". This person is characterised by the following:

- Does not reach or adequately defend a solution.
- Exhibits strong analysis skill, but appears to be "wishy-washy".
- Write papers that are too long and seem to ramble.
- Doesn't want to stop analysing.

24.6.1.2.2.4 Pragmatic performer

A pragmatic performer is a person who is characterised by the following:

- Objectively considers alternatives before reaching conclusions.
- Focuses on pragmatic solutions.
- Incorporates others in the decision process and/or implementation.
- Views task as finished when a solution/decision is reached.
- Gives insufficient attention to limitations, changing conditions, and strategic issues.
- Sometimes comes across as a "biased jumper", but reveals more complex thinking when prompted.

24.6.1.2.2.5 Strategic revisioner

A strategic revisioner is a person who is characterised by the following:

- Seeks continuous improvement/lifelong learning.
- More likely than others to think "out of the box".
- Anticipates change.
- Works toward construction knowledge over time.

24.6.1.3 Category 3: Individual traits

These are the traits providing the skills to communicate with, work with, lead and influence other people, ethics, integrity, etc. These traits include communications, personal relationships, team playing, influencing, negotiating, self-learning, establishing trust, managing, leading, emotional intelligence (Goleman, 1995), and more (Covey, 1989; Frank, 2010; ETA, 2010). These traits may be selected to suit the role of the systems engineer in the organisation and assessed in the way that the ETA industry standard competency models assess those traits (ETA, 2010). There is no need to reinvent an assessment approach.

24.6.2 The horizontal dimension

The horizontal dimension provides a way to assess the competence of a person in each broad area of the vertical dimension against the levels of increasing ability. The nine competency models discussed above defined proficiency in different ways and in different ranges as shown in Table 24-3. For example, Dreyfus and Dreyfus (Dreyfus and Dreyfus, 1986) quoted by Ennis (Ennis, 2008) describe levels of proficiency that include novice, experienced beginner, practitioner, knowledgeable practitioner, expert, virtuoso, and maestro. From the novice who is focused on rules and limited or inflexible in their behaviour to the individual who is willing to break rules to provide creative and innovative solutions to business problems. A way to encompass the existing approaches of assessing systems engineers needed to be developed.

Systems engineers should be innovators, problem formulators and solvers. Gordon et al. provided a way to identify the difference in cognitive skills between innovators, problem formulators, problem solvers and imitators (Gordon G. et al., 1974). The difference is based on:

- Ability to find differences among objects which seem to be similar,
- Ability to find similarities among objects which seem to be different

Table 24-4 Factors conducive to innovation

Ability to find **similarities** among objects which seem to be **different**	HIGH	Problem solvers	Innovators
	LOW	Imitators/ Doers	Problem Formulators
		LOW	HIGH

	Ability to find **differences** among objects which seem to be **similar**

The differences in the 'ability to find …' leads to the different type of persons shown in Table 24-4 (Gordon G. et al., 1974). For example, problem formulators score high in ability to find differences among objects which seem to be similar, and problem solvers score high in ability to find similarities among objects which seem to be different.

Anecdotal evidence (observation and experience) indicated that within the multichotomy of systems engineering there appeared to be five types of systems engineers[150] (Section 23.3) which may be mapped into Table 24-4 as shown in Table 24-5.

Table 24-5 Mapping Types into abilities

Ability to find **similarities** among objects which seem to be **different**	HIGH	Problem solvers **(Type III)**	Innovators **(Type V)**
	LOW	Imitators/Doers **(Type II)**	Problem Formulators **(Type IV)**
		LOW	HIGH
		Ability to find **differences** among objects which seem to be **similar**	

This Chapter has not attempted to map the types into roles or job titles because roles and titles vary with organisations. For example a 'lead systems engineer' in one organisation may be performing the same role as a person with the title 'senior systems engineer' or 'engineering specialist' in another organization. Moreover, these five types exist in other disciplines which would allow for the application of the framework in those disciplines by changing the knowledge components.

A two-dimensional CMMF showing the assessment of the competency in increasing levels of competency (Type I to V) in the three categories dis-

[150] The terminology of the 'types' once explained seems to resonate with the audience. The terminology has been adopted into common usage in the systems engineering vocabulary in Singapore and in Israel after it was introduced in a workshop in 2010.

cussed in Section 24.6.1 is summarised in Table 24-6. Assessment of knowledge, cognitive skills and individual traits is made in ways already practiced in the psychology domain and do not need to reinvented by systems engineers. Where knowledge is required at the conditional level, it includes procedural and declarative. Similarly, where knowledge is required at the procedural level, it includes declarative knowledge.

24.7 Benchmarking the nine competency models

Each of the three categories contain some competencies that are common to all systems engineers and some competencies that apply to specific roles in specific domains in specific phases of the system lifecycle in specific organisations. Each competency model thus contains information which can be allocated into these three categories and allows them to be subjectively compared or benchmarked as shown in Table 24-7. Findings from this comparison, based on the published literature, include:

- The number of levels of proficiency differs between competency models.
- The definition of the ability for a level of proficiency differs between competency models.
- The lack of competencies in the implementation domain in all nine competency models examined. However, it is fair to say that some of the models do consider the culture in which the systems engineering is taking place.

24.8 Future research

This section discusses the following future research opportunities.

- Identifying gaps in existing competency models
- Using the CMMF as a competency model.

24.8.1 Identifying gaps in existing competency models

The existing competency models seem to have been populated based on observing the role of the systems engineer, namely what systems engineers do in the workplace, and researching the literature for additional requirements. These competency models may suffer from errors of omission because the development methodology does not include a validation function to determine if something that should be done is not being done (and the effect of that lack may not show up for some months or even years). Indeed, this research has identified an error of omission in all of the nine competency models studied, namely the lack of competencies in the implementation domain. In addition, benchmarking used alone produces followers, not

leaders. Benchmarking should be used only as a check to make sure your competency model[151] is not lacking some necessary competency.

A useful tool for identifying both errors of commission and errors of omission is workflow analysis. Workflow analysis is a methodology that can be used to observe existing workflows and also to develop a conceptual reference model. The conceptual reference competency model activities can then be compared to those being observed and any missing functions identified and initiated. A useful conceptual reference model is the Generic Reference Model (Hitchins, 2007), with its mission management, resource management, viability management, behaviour management and form management sub-models.

A useful tool for mapping the activities in the workflow and hence the competencies needed to perform those activities in an objective manner is the HKMF shown in Figure 21-3. The activities that need to be performed in each area of the framework are identified in the form of descriptions, scenarios, use cases, vignettes, concepts of operations[152], etc. For example, consider the requirements engineering activities in Layer 2 Area 2B of the HKMF, these activities include requirements elicitation and elucidation. The appropriate competencies and proficiencies for the type of person performing the activity in each of the scenarios can then be determined via any one of the many methods used in the human resources domain.

24.9 Using the CMMF as a competency model

In order for an organization to use the CMMF, the contents of each of the three categories must be determined and the CMMF populated. If the organization already has a competency model then the competencies need to be transferred from the organisations' competency model into the appropriate areas in the CMMF. If the organization does not have a competency model and wishes to develop one, then the CMMF allows standardization of groupings which helps identify both errors of commission and errors of omission. However, before developing a competency model, a cost-benefit trade-offs should be performed since the amount of effort will depend on the level of detail required. The development effort should be for a model that will be useful, not something that will keep the human resource department busy.

[151] Or anything else you are creating and wish to benchmark.
[152] The terminology varies but the concept is the same.

Table 24-6 A Competency Model Maturity Framework (CMMF) for Systems Engineers

	Type I	Type II	Type III	Type IV	Type V
Category 1: Knowledge areas					
Systems engineering	Declarative	Procedural	Conditional	Conditional	Conditional
Problem domain	Declarative	Declarative	Conditional	Conditional	Conditional
Implementation domain	Declarative	Declarative	Conditional	Conditional	Conditional
Solution domain	Declarative	Declarative	Conditional	Conditional	Conditional
Category 2: Cognitive characteristics					
Holistic Thinking					
Descriptive (8)	Declarative	Procedural	Conditional	Conditional	Conditional
Prescriptive (1)	No	No	Procedural	No	Conditional
Critical Thinking	Confused fact finder	Perpetual analyser	Pragmatic performer	Pragmatic per-former	Strategic re-visioner
Category 3: Individual traits (typical sample) organisational specific not shown					
Communications	Needed	Needed	Needed	Needed	Needed
Management	Not needed	Needed	Needed	Needed	Needed
Leadership	Not needed	Not needed	Needed	Needed	Needed

Table 24-7 Comparison of competency models

Assessment approach	KSA's	INCOSE CSEP Exam	SECF	CEST	SECT	NASA 2010	JPL SEA	MITRE	NDIA
Category 1: Knowledge									
Systems Engineering	Yes [1]	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Solution domain	Implied	No	No	No	Yes	Yes	Yes	Yes	No
Implementation domain [3]	Implied	No	Partial [4]	No	No	Partial [4]	Partial [4]	Partial [4]	No
Problem domain	Implied	No	No	Yes	Yes	Yes	Implied	Yes	Yes
Category 2: Cognitive skills	Yes	No	Yes	Yes	Yes	Implied	Yes	Yes	Yes
Category 3: Individual Traits [5]	Yes	No	Unclear	Yes	Yes	Yes	Yes	Yes	Yes
Increasing levels of proficiency	No	No (Pass/Fail)	Partial [2]	No	Yes	Yes	No	Yes	No

Notes to Table 24-7**.**

In several instances, the various ways in which the competency models describe the competencies made populating this table a subjective exercise.

The use of the word 'Yes' should be read with the understanding that each competency model identifies a different set of knowledge in each of the knowledge area rows in the table.

[1] Subjective approach, knowledge seems to be dependent on situation, no objective reference for validating characteristics.

[2] Lowest level is in a different dimension to remaining levels

[3] Systems engineering tools have been allocated to the systems engineering knowledge area rather than to implementation domain.

[4] Does contain knowledge about the culture of the organisation is which the systems engineering is taking place.

[5] These will depend on the requirements for the postion.

Candidates must qualify at the appropriate proficiency level in all three categories to be recognised as being competent that competency level. While examination questions can require the respondent to use conditional knowledge, the successful application of conditional knowledge in the real world must be directly demonstrated by results documented in the form of KSAs supported by awards, letters and certificates of appreciation from third parties (e.g. employers, customers, etc.). The assessment could thus be in two parts, one part by examination for the lower levels, the second by a portfolio demonstrating successful experience for the higher levels. This model is followed by the INCOSE Certification Program in which the ESEP is awarded on the basis of a portfolio. Other higher level qualifications based on portfolios are awarded by the Association for Learning Technology (ALT) and the Institution of Engineering and Technology (IET) in the UK namely Certified Member of the ALT (CMALT) and Fellow of the IET (FIET).

Assessment of a candidate is simple in concept as follows.

24.9.1 The cognitive skills and individual traits

Knowledge of the HTPs is assessed as declarative, procedural and conditional (Woolfolk, 1998). Examination questions may be written to require the respondent to demonstrate the different types of knowledge. Ways of assessing the degree of critical thinking have been described by Wolcott and Gray (Wolcott and Gray, 2003) and are used herein. The appropriate individual traits are assessed as being 'needed' or 'not needed' at a specific level of ability.

24.9.2 The systems engineering, implementation, problem and solution domain knowledge

The knowledge is also assessed as being declarative, procedural and conditional (Woolfolk, 1998). The question then arises as what is the knowledge to be? As stated above, consensus on the contents of a 'standard' SEBoK would be difficult to achieve across organisations and domains if it were to be based on the role of the systems engineer. That the knowledge competency is situational rather than generic does not stop the CMMF being populated by organisations needing competency assessments for their personnel working in their environment on their projects.

24.10 Summary and conclusions

The Chapter began with a discussion on the need for competent systems engineers. The Chapter then discussed the role of systems engineers in the workplace providing examples of a number of points of view as to what those roles are and alludes to the difficulty of gaining a generic consensus on the nature of the role of the systems engineer. At that point focus of the Chapter changed to ways of assessing the competency of systems engineers using terminology from the domain of cognitive psychology rather than inventing new systems engineering terminology for existing concepts. The Chapter continued with a brief examination and discussion of nine different competency models and showed that each competency model seems to have been designed for a different purpose in different domains, times and places.

The Chapter then introduced a two-dimensional CMMF that can be used for both benchmarking existing competency models and as a competency model by those organisations that do not yet have a competency model and wish to develop one. The nine competency models were benchmarked using the CMMF and one significant finding from this comparison is the lack of competencies in the implementation domain in all nine-competency models studied. The Chapter concluded with suggestions for future research.

2010

25 Unifying the different systems engineering processes

"It ain't what you don't know that gets you into trouble. It's what you know for sure that just ain't so." - Mark Twain 1835-1910.

Teaching the systems engineering process is difficult because of the contradictory and confusing process information in the literature as well as the overlap between the systems engineering process and the problem solving process as well as the confusion between the systems engineering process and the system lifecycle. This Chapter resolves the conflict and confusion and documents two separate meta-systems engineering processes; a 'planning' process that produces the planning documents and a 'doing' process in which the plan is implemented. The result of this research is a meta-model of the two systems engineering processes that not only facilitate teaching by showing that all documented systems engineering processes are subsets of the meta-processes, but also show how agile systems engineering, lean systems engineering and evolutionary acquisition all fit together in an integrated manner.

25.1 Introduction

In teaching systems engineering it has been observed that students that come into the class knowing some systems In teaching systems engineering it has been observed that students that come into the class knowing some systems engineering come out of the class knowing a little more systems engineering, while students that come into the class not knowing systems engineering, come out of the class not knowing it a little less. Reflection on this situation has indicated that there may be ways to improve the way the systems engineering process is taught, including:

1. Pointing out the myth of the systems engineering process.

2. Explaining the overlap between some versions of the systems engineering process and the problem solving process.
3. The way iteration of/in the systems engineering process is taught.
4. The misuse of functional diagrams to represent processes.

Consider these four points.

25.2 The myth of the single systems engineering process

According to the US DOD 5000 Guidebook 4.1.1, "The successful implementation of proven, disciplined SEPs results in a total system solution that is--

Robust to changing technical, production, and operating environments;
Adaptive to the needs of the user; and
Balanced among the multiple requirements, design considerations, design constraints, and program budgets".

"*A single process, standardizing the scope, purpose and a set of development actions, has been traditionally associated with systems engineering" (Arnold, 2000) citing (MIL-STD-499B, 1993) and (IEEE 1220, 1998). However*, there is no single widely agreed upon SEP since over the years, the SEP has been stated in many different ways, including:

- The (EIA 632, 1994) and (IEEE 1220, 1998) processes shown in Figure 21-13 and Figure 25-1;
- Lists of processes in ISO/IEC 15288 (Arnold, 2002);
- The waterfall process (Royce, 1970);
- The V model version of the process;
- The spiral, incremental and evolutionary models;
- System Lifecycle functions (Blanchard and Fabrycky, 1981) shown in Figure 25-2;
- State, Investigate, Model, Integrate, Launch, Assess and Re-evaluate (SIMILAR) (Bahill and Gissing, 1998) shown in Figure 25-3;
- The basic core concepts accepted by most systems engineers (Mar, 2009b);
- A systems engineering approach to addressing a problem (Hitchins, 2007).

Consequently, given the conflicting and contradictory information in the various versions of the systems engineering process, the systems engineering process concept is difficult to explain and, teaching has focused on using the waterfall and V models since, while not representative of the real world, they are simple to explain (Biemer and Sage, 2009) pages 152 and 153).

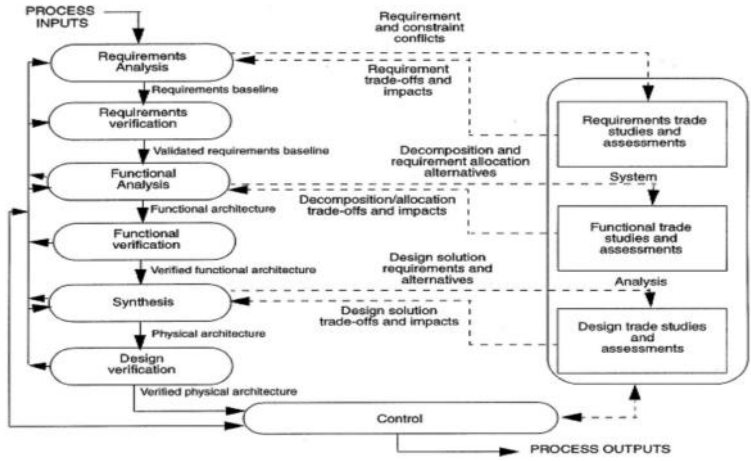


Figure 4—Systems engineering process (SEP)

Figure 25-1 IEEE 1220 Systems Engineering Process

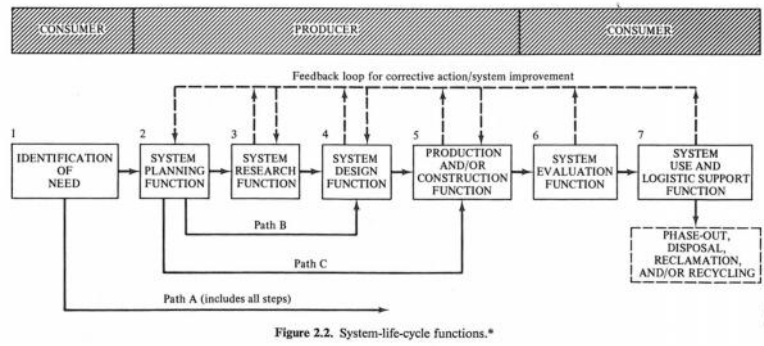


Figure 2.2. System-life-cycle functions.*

Figure 25-2 System Lifecycle functions (Blanchard and Fabrycky, 1981)

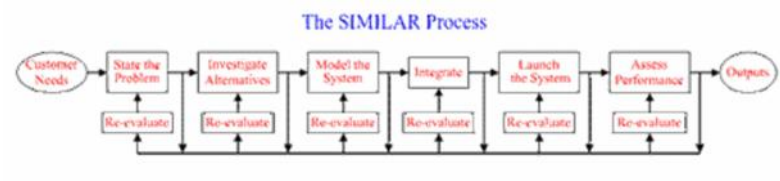


Figure 25-3 The SIMILAR process (Bahill and Gissing, 1998)

The key insight to understanding the reason for the variety of systems engineering processes may lie with Biemer and Sage who state that "*the systems engineer creates a unique process for his or her particular development effort*" (Biemer and Sage, 2009) page 153). Consider each published version of the systems engineering process [153] as the **unique process created for their particular development effort** by someone or some group at some point in time, at some point in the system lifecycle, in the context of what they defined as a systems engineering problem and subsequently documented as their systems engineering process.

Looking for patterns in the various versions of the systems engineering process listed above as well as others in the literature, one can identity versions that:

- focus on early stage systems engineering where the problem is explored and conceptual solutions developed;
- focus on engineering the system and realizing the solution;
- focus on both aspects.

A process is a sequence of activities. The systems engineering process takes place in the context of the HKMF shown in Figure 21-3. HKMF Column A contains activities that address the initial problem and conceptual solution (Kasser, et al., 2009). Columns B, C, D, E and F contain the activities that realize the solution.

From the Big Picture perspective, there seem to be two interdependent systems engineering processes:

- **The traditional 'doing' systems engineering process** in which Layer 2 systems engineering is performed. This is the unique systems engineering process which is constructed for the realization of a specific system. The activities performed in the unique systems engineering process will depend on the problem-identification-solution-realization activities that have and have not been done at the time the unique systems engineering process is constructed.
- **The planning systems engineering process;** the process used by the systems engineer to create the unique systems engineering process. When designing/planning the unique systems engineering process for the realization of a system, systems engineers use knowledge based on experience and the activities functions and processes which can be found in the processes and Standards listed above and in the literature. This planning process is a problem solving activity, consequently it ought to, and does, map into the problem solving process.

[153] In a Standard or in a textbook.

25.3 The overlap between some versions of the systems engineering process and the problem solving process

Mar stated, "*Most systems engineers accept the following basic core concepts*[154]:

1. *Understand the whole problem before you try to solve it*.
2. *Translate the problem into measurable requirements*.
3. *Examine all feasible alternatives before selecting a solution*.
4. *Make sure you consider the total system lifecycle. The birth to death concept extends to maintenance, replacement and decommission. If these are not considered in the other tasks, major lifecycle costs can be ignored*.
5. *Make sure to test the total system before delivering it*" (Mar, 2009b).

Two outlines of generic problem solving processes are shown in Table 25-1. They are the Global Development Research Center (GDRC) version which covers the problem identification-solution identification steps (GDRC, 2009) and the Office of Vocational and Adult Education (OVAE) (OVAE, 2005) which also covers the solution realization and evaluation steps. If one compares these examples of the problem solving process with the functions (Blanchard and Fabrycky, 1981) shown in Figure 25-2, the SIMILAR process (Bahill and Gissing, 1998) shown in Figure 25-3 and assertion by (Mar, 2009a), it can be seen that while the aggregation of activities into the steps are different and the level of detail in each step is different, these versions of the SEP seem to be the same as the problem solving process.

Table 25-1 Two versions of the problem solving process

GDRC, 2009	OVAE, 2005
1. Problem Definition	1. Identify and Select the Problem
2. Problem Analysis.	2. Analyse the Problem
3. Generating possible Solutions.	3. Generate Potential Solutions
4. Analysing the Solutions.	4. Select and Plan the Solution
5. Selecting the best Solution(s).	5. Implement the Solution
6. Planning the next course of action (Next Steps)	6. Evaluate the Solution

The confusion between the systems engineering process and the problem solving process can be resolved by recognizing that from the problem

[154] Of systems engineering, Author's interpretation.

solving perspective, a man-made system is realized as a solution to a problem. As such, the parts of the systems engineering process that takes place in Columns A and C of the HKMF constitute problem-solving processes.

25.4 The way iteration of/in the systems engineering process is taught

The iterative nature of systems engineering has sometimes been taught using the egg diagram shown in Figure 21-13 (EIA 632, 1994) as being applicable to each phase of the system lifecycle in the manner shown in Figure 25-4. Since the contents of the egg are expressed in Layer 2 realization language it is difficult to get the concept of iteration across to the students is because the words have incorrect meanings in the other phases and layers. In addition, this teaching approach only addresses a part of the iterative nature of systems engineering which includes the following four types of iteration.

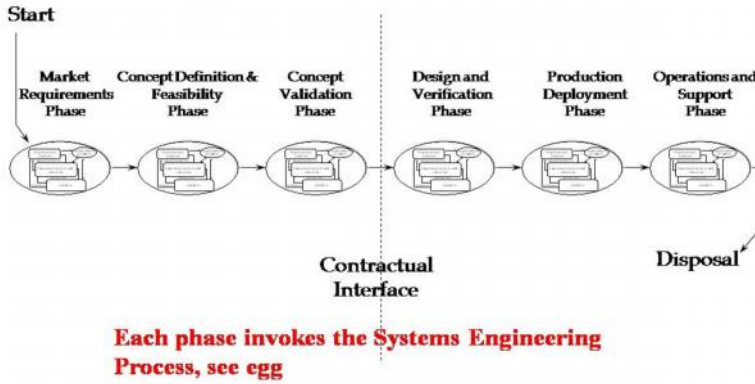


Figure 25-4 A Typical System Lifecycle (UNiSA, 2006)

1. Iteration inside an area of the HKMF.
2. Iteration across a row of the HKMF.
3. Iteration in a column of the HKMF.
4. Iteration of a number of system lifecycles in series.

Consider each of them.

- **Iteration inside an area of the HKMF.** This type of iteration takes place when a process is repeated during the production of a product. It is generally drawn as a circular sequence of activities. Two examples are:

  1. The iterative part of the process for producing a document. The system engineer produces a version of a document, circulates it for comment, receives comments, incorporates the comments in the document and circulates the document for further comment

(Kasser, 1995) pages 158-160). This writing-reviewing-update loop takes place until the criterion for terminating the loop is reached. Examples of such termination criteria include the document is complete, or the scheduled date for delivering the document has been reached.

2. The design activities which take place in Columns A.2 and A.3 of the HKMF.

- **Iteration across a row of the HKMF.** This type of iteration takes place when the same sequence of activities is performed in more than one column of the HKMF. An example is the design process which takes place at the conceptual level[155] in Column A and at the realization level in Column C.
- **Iteration in a column of the HKMF.** Systems engineering is a problem solving discipline and while the types of problems that are found in each column are different, the tools and techniques used to solve them will be different but the problem solving approach will most likely be the same. For example,
    - A Layer 3 situation dealing with human issues may require an adaption of an appropriate methodology such as the Soft Systems Methodology (Checkland and Scholes, 1990) while a Layer 2 situation in the same column applying to a different project may require the use of Quality Function Deployment (QFD) (Clausing, 1994), queuing theory, linear programming or some other mathematical approach.
    - In government acquisitions in Layer 2, the preferred implementation option determined in Column A is often to outsource the realization and proving phases to a contractor. In such a situation, Column A contains the activities that would produce:
    - the acquisition plan;
    - the tender or request for proposal;
    - the tender or proposal evaluation and selection of development contractor;
    - the contract for the realization and proving phases in Columns B to F.

[155] To complicate the situation, the conceptual design process in Column A also iterates a realization design process to the extent needed to show that the conceptual design is feasible and to identify any risks associated with realizing that design (e.g. technological, schedule, etc.).

- If the disposal method for a system has not been predetermined, Column H may cycle though Columns A to F for the system disposal project.

- **Iteration of a number of system lifecycles in series.** This type of iteration has a number of names including evolutionary acquisition, sequential software Builds and the cataract process (Chapter 13). Two ways of thinking about the nature of this type of iteration are

  1. To consider the waterfall with its activities mapped into Columns A to F as the first iteration through the SEP. Changes are requested in the performance of the system during Column G. Configuration control allocates a set of changes to an upgraded version and an iteration of the waterfall activities mapped into Columns A to F take place for each new version of the system.
  2. To use two waterfalls, one placed below the other such that the end of the first Waterfall is in line with the start of the lower waterfall as shown in Figure 25-5

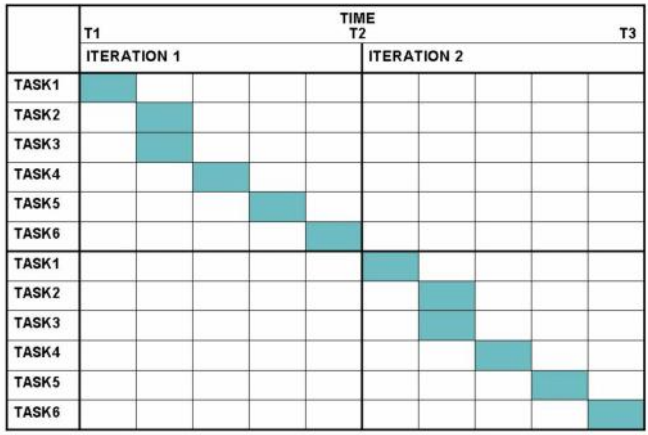


Figure 25-5 Gantt chart representation of iteration

25.5 The misuse of functional diagrams to represent processes

Another teaching difficulty is built into the graphical representations used in teaching, Consider typical examples; the EIA 632 version (EIA 632, 1994) shown in Figure 21-13, the IEEE 1220 version (IEEE 1220, 1998) shown in Figure 25-1, the SIMILAR process shown in Figure 25-3 (Bahill and Gissing, 1998) and the system lifecycle functions shown in Figure 25-2 (Blanchard and Fabrycky, 1981). These figures do not show processes they show func-

tions and the relationship between the functions. Looping back from the end of a sequence of functions to the start is generally added in functional drawings by adding a feedback arrow from the output of a function to the input of a previous function. Processes take time to implement and time does not flow backwards so there shall not be feedback lines in a process. Systems engineers process architect (Chapter 19) or create a unique systems engineering process for realizing a system (Biemer and Sage, 2009) page 153). The resulting systems engineering process is depicted in the form of Gantt or PERT style charts not in drawings containing feedback loops. For example, when a sequence of six tasks need to be repeated (iterated) the process shall be drawn as a Gantt chart showing a second set of activities right-shifted as shown in Figure 25-5 instead of as a flow chart with a feedback link from the end of Task 6 to the start of Task 1.

25.6 The common systems engineering process

Systems engineering is a problem solving activity as discussed above. Systems engineers process architect (Chapter 19) or create (Biemer and Sage, 2009) page 153) a unique systems engineering process for realizing a system. The planning process they use to create the unique systems engineering process for realizing a system should be common to all systems engineering activities. If the systems engineering activity is considered as a project, then a common meta- systems engineering process can be created by combining the Hitchins (Hitchins, 2007) page 173) and Mar approaches into the following 10-step sequence that joins the problem solving process and the solution realization process (Mar, 2009b):

1. Plan the project.
2. Explore/survey the problem space.
3. Conceive at least two feasible ways to tackling the problem by solving, resolving, dissolving or absolving it (Ackoff, 1999) page 115). If the problem is to be absolved, then proceed directly to Step 10.
4. Identify ideal selection criteria for evaluation of the feasible ways of addressing the problem.
5. Perform trade-offs to determine and select the best way of addressing the problem.
6. Fine tune selected option.
7. Formulate strategies and plans to realize preferred option.
8. Realize preferred option.
9. Verify that preferred option tackled the problem.
10. Terminate the project.

Notes:

a) Step 1 may sometimes take place in less detail before the project begins to determine that there is a need for the project and to allocate an initial set of resources.
b) Step 1 should include a review of best practices lessons learned from previous and similar projects to determine what worked and what did not work (the both the process and product domains) in the context of the similar projects, and the nature of the differences between the similar projects and this one.
c) In practice, Steps 3 and 4 may be conducted in parallel, not sequentially.
d) Iteration may take place as discussed above.
e) Step 10 includes documenting the lessons learned from the project.

Restating this meta-problem-solving-solution-realization process as a 'planning' process to create the unique systems engineering process, the wording would be:

1. Plan the project that will create the required planning documentation for the unique systems engineering process that will realize the solution system.
2. Explore/survey what needs to be done.
3. Conceive at least two feasible systems engineering processes.
4. Identify ideal selection criteria for evaluation of the systems engineering processes.
5. Perform trade-offs to determine and select the best systems engineering process.
6. Fine tune selected systems engineering process.
7. Formulate strategies and plans to realize preferred systems engineering process.
8. Document preferred systems engineering process using activities as building blocks in the appropriate planning documentation.
9. Obtain stakeholder consensus that the planned unique 'doing' systems engineering process can realize the solution system.
10. Terminate the project. This step begins the transition from 'planning' to 'doing'.

25.7 Lean and agile systems engineering

Lean and agile systems engineering do not need special processes and treatment when designing the unique systems engineering process to realize a system since:

- **Lean systems engineering** takes place when the unique systems engineering process designed to realize the system does not contain any non-productive activities. Since non-productive activities are wasteful, lean systems engineering should be the norm.

- **Agile systems engineering** takes place when the system lifecycle is short enough to deliver a solution in time to deal with the problem. In a situation where the problem changes during the time the solution is being developed, the systems engineering process should iterate a number of system lifecycles to provide timely solutions to the changing problems. This is the evolutionary paradigm albeit with a shorter than usual lifecycle time and is not a special case of systems engineering.

25.8 Summary

Starting with the observation that in teaching systems engineering students that come into the class knowing some systems engineering come out of the class knowing a little more systems engineering, while students that come into the class not knowing systems engineering, come out of the class not knowing it a little less, the Chapter discussed four problems associated with the way the systems engineering process is taught and clarified some of the confusion and contradictory information associated with current teaching approaches.

25.9 Conclusions

This research has shown that the single systems engineering process is a myth due to the way the systems engineering process is currently taught. From the Big Picture perspective, there seem to be two interdependent parts of a meta-systems engineering process, one for 'planning' and one for 'doing' or realizing the solution system:

- The unique 'doing' systems engineering process is constructed for the realization of a specific system. When designing the unique systems engineering process for the realization of a system in the areas of the HKMF to be inhabited by the unique systems engineering process, systems engineers use knowledge based on experience and the activities functions and processes which can be found in the processes and Standards listed above and in the literature as building blocks. The activities to be performed in the unique systems engineering process will depend on the work that has and has not been done at the point in the system lifecycle in which the process is constructed.
- The second part of the meta-systems engineering process is the 'planning' process used by the systems engineer to create the unique systems engineering process. Since this process is a problem solving activity, it ought to, and does, map into the problem solving process.

  This research has also:

- Shown that agile systems engineering and lean systems engineering are not special cases of systems engineering and should be the norm.
- Clarified the conflicting and contradictory information in the various versions of the systems engineering processes by viewing them as different unique subsets of the meta- systems engineering process appropriate to their situation.
- Determined that the columns in the HKMF may need adjusting. The HKMF was developed from the (EIA 632, 1994) and (IEEE 1220, 1998) perspectives and rolled up the early phases of the system lifecycle into a single phase labelled 'needs identification'. However, during the course of developing a framework for a SEBoK (Sections 12, 21 and 23) Column A, the 'needs identification' phase had to be expanded into three subphases to properly address the problem exploration and solution determination phases inside Column A.
- Raised the need for further research to explore aligning the columns of the HKMF (phases of the system lifecycle) with the appropriate steps of the problem solving process perhaps by expanding Column A and rolling up Columns C, D, E and part of F into a single 'realization' column. This change would keep the number of columns manageable and should further facilitate teaching about the SEP.

Since the unique SEP is constructed from 'building blocks' described in the Standards, text books and other literature, further research should be performed to create standard set of building blocks for the systems engineering and non-systems engineering activities in each area of the HKMF. Use of these process building blocks would be similar to the way electronic engineers use digital integrated circuits to create digital circuits. The building blocks would have a standard format. Inclusion of the building blocks and a software agent to verify that the blocks are linked together (in a similar manner to the way requirements management tools monitor traceability) would be a useful way of adding intelligence to project management tools.

2010

26 Seven systems engineering myths and the corresponding realities

"It ain't what you don't know that gets you into trouble. It's what you know for sure that just ain't so." - Mark Twain 1835-1910.

This Chapter states that systems engineering is a discipline characterized by debates based on subjective opinions, with participants talking past each other, a lack of listening and a number of myths. The opinions expressed in this Chapter are based on some of the findings from research into the nature of systems engineering that began in 1994 and have been documented in this book. The Chapter discusses seven myths of systems engineering and shows the nature of the myth and the reality, and explains how and why each myth arose.

26.1 Introduction

In the second session of the Academic Forum at the 2009 International Symposium in Singapore, the state of systems engineering as a discipline was compared to the state of:

- electrical engineering before Ohm's law was postulated,
- electrical engineering before Maxwell's equations were stated, when engineers built motors by winding coils but had no theory upon which to predict the behaviour of the motor before powering it up for the first time
- chemistry before the periodic table of elements was discovered, and
- medicine in the 1800's before medical science provided a theory of why some medications work and why some don't.

Namely systems engineering is in its early stages. A discipline in these stages is characterized by debates based on subjective opinions, with participants talking past each other, a lack of listening, contradictory and confusing information and a number of myths. This Chapter addresses some of those myths and the opinions expressed in this Chapter are based on find-

ings from research into the nature of systems engineering that began in 1994. These partial findings are grouped herein as seven myths of systems engineering. The Chapter shows the nature of each myth, the reality, and explains how and why each myth arose. The myths discussed are:

- Myth 1: There are Standards for systems engineering.
- Myth 2: The "V" model of the systems engineering process
- Myth 3: Follow the systems engineering process and all will be well
- Myth 4: Complexity needs new tools and techniques
- Myth 5: Systems of systems are a different class of problem and need new tools and techniques
- Myth 6: Changing requirements are a cause of project failure so get the requirements up front.
- Myth 7: The systems engineering process.

Consider each myth and corresponding reality.

26.2 Myth 1: There are Standards for systems engineering

26.2.1 The myth

MIL-STD 499, EIA 632, IEEE 1220 and ISO/IEC 15288 (MIL-STD-499, 1969; EIA 632, 1994; IEEE 1220, 1998; Arnold, 2002) are commonly thought of as systems engineering standards.

26.2.2 The reality

The reality is that the approved Standards used in systems engineering cover systems engineering management and the processes for engineering a system; that is they do not seem to actually apply to systems engineering. Thus:

- Mil-STD-499 covers systems engineering management (MIL-STD-499, 1969).
- Mil-STD-499A covers engineering management (MIL-STD-499A, 1974) dropping the word 'systems' from the title.
- The draft (MIL-STD-499B, 1993) and MIL-STD-499C (Pennell and Knight, 2005) Standards contain the words "systems engineering" in their titles but the Standards were never approved.
- ANSI/EIA-632 covers processes for engineering a system (ANSI/EIA-632, 1999).
- The IEEE 1220 Standard is for the application and management of the systems engineering process (IEEE 1220, 1998).
- The ISO/IEC 15288 Standard lists processes performed by systems engineers (Arnold, 2002) and hence may be considered as being applicable to the role of the systems engineer rather than to the activities known

as systems engineering. In addition, many of the activities in ISO/IEC 15288 also overlap those of project management.

The lack of coverage of early stage systems engineering in the standards and the Capability Maturity Model Integration (CMMI) was discussed in Section 23.3.

Studies have shown that the cost of a system is determined in its early stages. A typical example shown in Figure 26-1 is a Defense Acquisition University study quoted in the INCOSE systems engineering handbook (Haskins, 2006b) page 2.6 of 10). The figure shows that 70% of costs of a system are committed by activities in the early stage of systems engineering; yet the Standards ignore those early stages and so seem to be focused on the wrong end of the system lifecycle.

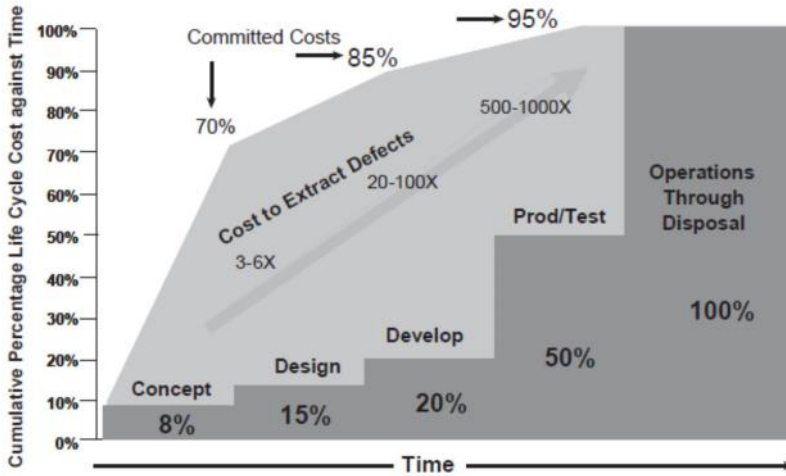


Figure 26-1 When costs are committed lifecycle

The DODAF was designed to be used to "*provide correct and timely information to decision makers involved in future acquisitions of communications equipment*" (DoDAF, 2004). Volume i contains 83 pages of definitions, guidelines, and background; volume ii contains 249 pages of product descriptions. The Deskbook contains 256 pages of supplementary information to framework users. The underlying data model comes with 696 pages and over 1200 data elements. The degree of micromanagement is phenomenal and expensive. Even a limited subset of the required information took 45,000 man-hours to produce (Davis, 2003). A chart mapping the degree of micromanagement in the standards over time (as measured by the thickness

of the document) is shown in Figure 26-2 which roughly corresponds to the same curve as the cost to fix a defect as a function of the time the defect is discovered[156]. As stated above, the early stages of systems engineering to the left of the vertical axis in Figure 26-2 is not covered by the standards. While DOD 5000 (DOD 5000.2-R, 2002) pages 73-74) does call out some of the early stage activities, those activities are called out as part of the separate seemingly independent CAIV process which takes place before the DOD 5000.2-R systems engineering process begins. CAIV is to be performed by Integrated Product and Process Development (IPPD) activities which involve organizing the different functions to work concurrently and collectively so that all aspects of the lifecycle for the various concepts are examined and a balanced concept emerges (DOD IPPD, 1998). In broad terms, the objectives of the IPPD concept exploration phase are fourfold:

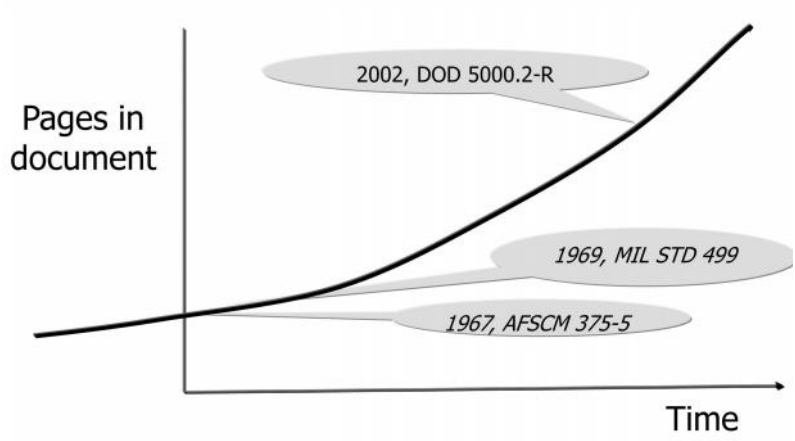


Figure 26-2 Increase in pages in Standards over time

1. to perform concept studies to investigate different solutions,
2. to evaluate these different concepts,
3. to perform trade-off studies, and
4. to define the requirements for the remainder of the acquisition program.

So, the US DOD moved early stage systems engineering out of systems engineering into CAIV and the activities were to be performed by IPPD teams rather than by systems engineers. The DOD paradigm resulted in textbooks which comply with DOD 5000 (DOD 5000.2-R, 2002), pages 83-84) and consider requirements as one input to the systems engineering process (Martin, 1997) page 95), (Eisner, 1997) page 9), (Wasson, 2006) page 60).

[156] No connection between the items is implied.

Standards continue to appear yet we need to stop legislating processes, the micromanagement of processes and the production of lists of boxes to be ticked and start educating Type V systems engineers who can solve problems (Section 23.3).

26.3 Myth 2: The "V" model of the systems engineering process

26.3.1 The myth

The V diagram is often used as a description of the systems engineering process (Section 20.2). Consider the representation of the V model in Figure 26-3 (Caltrans, 2007) and the typical representation of the waterfall model shown in Figure 26-4. Note that if the last two boxes in the waterfall are moved up to the corresponding levels as the first two boxes in the waterfall, the result is a V.

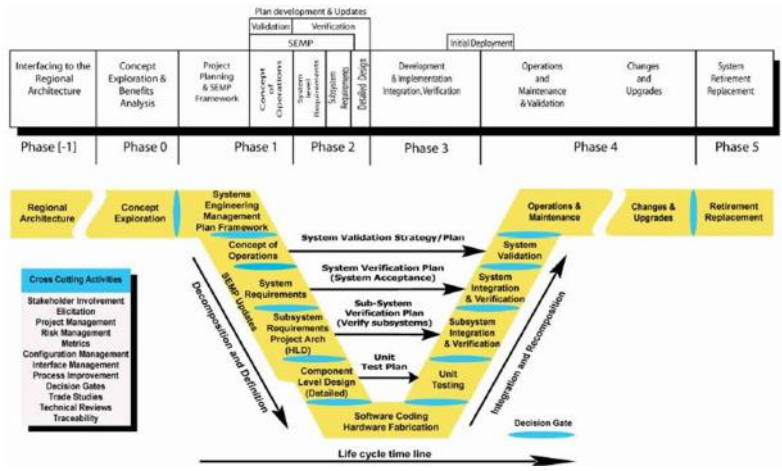


Figure 26-3 Example of the V Model (Caltrans, 2007)

26.3.2 The reality

The reality is that the V is the waterfall just drawn differently (Section 20.2)!

26.3.3 The dark side of the V

The use of the V view as a process model also perpetuates the following undesired practices.

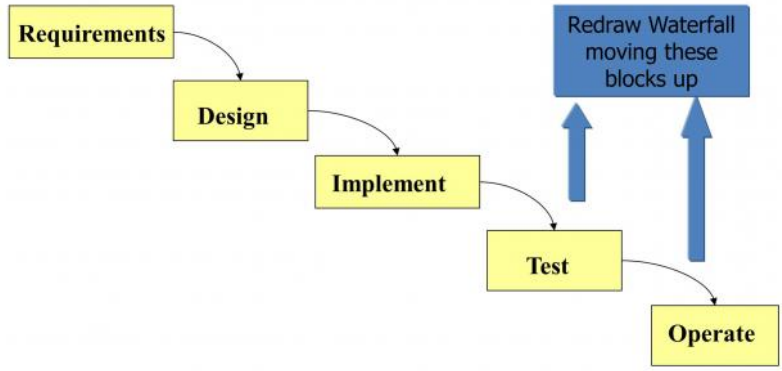


Figure 26-4 Redrawing the waterfall model

- Lack of prevention of defects.
- Failure to consider changes to customer needs during development of the solution system.

26.3.3.1 Lack of prevention of defects

When the V diagram is used in a simplistic manner to depict the relationship between development and T&E there seems to be no place in the diagram for the prevention of defects (Section 20.2). While the development team implements the system, the test team is busy planning the tests. A definition of a successful test is one that finds defects[157] (Myers, 1979). This is because if no defects are found, the result is ambiguous, because either there are no defects or the testing was not good enough to detect any defects. The lack of prevention of defects escalates costs. Deming wrote "*Quality comes not from inspection, but from improvement of the production process*" (Deming, 1986) page 29). He also wrote "*Defects are not free. Somebody makes them, and gets paid for making them*" (Deming, 1986) page 11). If the test team can identify defects to test for, why can't they hold a workshop or other type of meeting to sensitize the development team to those defects and hence prevent them from being built into the system? Such workshops in postgraduate courses at UMUC (1997-1999) and the UniSA (2000-2006) have sensitized students to the problems caused by poorly written requirements (Kasser, et al., 2003).

[157] As opposed to the goal of the system development team which is to produce a defect free system.

26.3.3.2 Failure to consider changes to customer needs during development of the solution system

The V is a redrawn waterfall as shown in Figure 26-4 and suffers from the same defect, namely lack of consideration of changes in customer needs. Some attempt however is sometimes made to include the effect of these changes by drawing two V's in series.

26.4 Myth 3: Follow the systems engineering process and all will be well

26.4.1 The myth

The myth of the single systems engineering process to was discussed in Chapter 25.

26.4.2 The reality

The reality in the literature is that excellence comes from people not process. The much quoted Chaos study (CHAOS, 1995) fails to mention process or lack thereof as a major contribution to project success or failure (Section 20.8). The literature is full of advice as to how to make projects succeed (Harrington, 1995; Peters, 1987; Peters and Austin, 1985; Rodgers, et al., 1993; Peters and Waterman, 1982) which in general tend to ignore process and focus on people. Systems engineers focus on developing processes for organizations – namely the rules for producing products. Companies don't want employees who can follow rules; they want people who can make the rules (Hammer and Champy, 1993) page 70). The contribution of good people in an organization was recognized in the systems engineering literature about 50 years ago, namely *"Management has a design and operation function, as does engineering. The design is usually done under the heading of organization. It should be noted first that the performance of a group of people is a strong function of the capabilities of the individuals and a rather weak function of the way they are organized. That is, good people do a fairly good job under almost any organization and a somewhat better one when the organization is good. Poor talent does a poor job with a bad organization, but it is still a poor job no matter what the organization. Repeated reorganizations are noted in groups of individuals poorly suited to their function, though no amount of good organization will give good performance. The best architectural design fails with poor bricks and mortar. But the payoff from good organization with good people is worthwhile"* (Goode and Machol, 1959) page 514). Excellence is in the person not the process. Again the focus should be on developing Type V engineer leaders (Section 23.3) rather than on developing more and more detailed processes.

26.5 Myth 4: Complexity needs new tools and techniques

Systems engineering has not delivered on its promise to meet the challenge of complexity as documented by Chestnut who wrote "*Characteristic of our times are the concepts of complexity, growth and change*" (Chestnut, 1965) page 1) and "*in a society which is producing more people, more materials, more things, and more information than ever before, systems engineering is indispensable in meeting the challenge of complexity*" (Chestnut, 1965) page vii). There is a growing dichotomy in the literature on the subject of complex systems.

26.5.1 The myth

The myth is represented in literature on the need to develop new tools and techniques to manage them. For example:

- Bar-Yam who proposed that "*complex engineering projects should be managed as evolutionary processes that undergo continuous rapid improvement through iterative incremental changes performed in parallel and thus is linked to diverse small subsystems of various sizes and relationships. Constraints and dependencies increase complexity and should be imposed only when necessary. This context must establish necessary security for task performance and for the system that is performing the tasks. In the evolutionary context, people and technology are agents that are involved in design, implementation and function. Management's basic oversight (meta) tasks are to create a context and design the process of innovation, and to shorten the natural feedback loops through extended measures of performance*" (Bar-Yam, 2003). Bar-Yam:
    - quoted the Chaos study (CHAOS, 1995) suggesting that the systemic reason for the challenged project is their inherent complexity. That might be one finding, however, the general finding from the Chaos study that the systemic reason for the challenged projects is poor management!
    - Cited own prior work "*for all practical purposes adequate testing of complex engineered systems is impossible*"
    - Suggested evolutionary process for engineering large complex systems.

26.5.2 The reality

The reality is represented in the literature on techniques such as aggregation which masks the underlying complexity to ensure that only the pertinent details for the particular situation to deal with the issues are considered. For example:

- Jenkins defined systems engineering as "*the science of designing complex systems in their totality to ensure that the component subsystems*

making up the system are designed, fitted together, checked and operated in the most efficient way" (Jenkins, 1969).

- Maier and Rechtin who recommend that the way to deal with high levels of complexity is to abstract the system at as high a level as possible and then progressively reduce the level of abstraction (Maier and Rechtin, 2000).

There seem to be two types of complexity as follows:

- **Real world complexity** - in which elements of the real world are related in some fashion, and made up of components. This complexity is not reduced by appropriate abstraction it is only hidden.
- **Artificial complexity** – arising from either poor aggregation (Maier and Rechtin, 2000) or elements of the real world that, in most instances, should have been abstracted out when drawing the internal and external system boundaries, since they are not relevant to the purpose for which the system was created. It is this artificial complexity that gives rise to complication in the manner of Rube Goldberg or W. Heath Robinson[158]. For example, in today's paradigm, complicated drawings are generated that contain lots of information[159] and the observer abstracts information as necessary from the drawings. The natural complexity of the area of interest is included in the drawings. Hence the system is thought to be complex.

Dealing with complexity means using abstraction and elaboration (Hitchins, 2003) pages 93-95) coupled with domain knowledge to develop an understanding of the situation, namely interrelationships among the system components and knowing which are pertinent to the situation and which can be safely ignored. For example, the space transportation system (space shuttle) and the international space station are both complex systems. However, when considering the problem of docking one to the other all aspects of the situation can be abstracted out except for the relative velocities, distance and alignments (yaw and pitch).

Perhaps the existence of the dichotomy is due to the observation that "*the classification of a system as complex or simple will depend upon the observer of the system and upon the purpose he has for considering the system*" (Jackson and Keys, 1984). Bar-Yam seems to drawing conclusions from poor engineering and management. He is correct in writing "*that for all practical purposes adequate testing of complex engineered systems is impossible*"; However the Continuum perspective indicates that Bar Yam's state-

[158] Cartoonists in the USA and UK who drew cartoons of complicated systems designed to perform simple functions.
[159] DoDAF OV diagrams can be wonderful examples of complexity.

ment only applies to the architectures in use today; there should be other architectures that would allow adequate testing. His suggestion for an evolutionary process has been applied to all types for systems since antiquity. The concept of establishing baselines and then using a "build a little, test a little" approach is well established in all areas of activity.

26.6 Myth 5: Systems of systems are a different class of problem and need new tools and techniques

There is a dichotomy on the issue similar to the dichotomy on complexity. The earliest reference to system of systems found in the literature was Jackson and Keys who wrote that a problem solver needs a methodology for [selecting the appropriate methodology for] solving a problem (Jackson and Keys, 1984) which has nothing to do with the use of the term in modern systems engineering.

26.6.1 The myth

Allison and Cook defined a system of systems as "*a system made up of elements that are not acquired or designed as a single system but are acquired over time and are in continuous evolution*" (Allison and Cook, 1998). They categorized system of systems are permanent, such as airlines and national Defence forces, and temporary, ephemeral or virtual examples of such as multi-national peace keeping forces and project teams. Cook stated that "*the term system of systems in its permanent sense has come to mean a set of interdependent systems evolving at different rates, each at a different phase of their individual system lifecycles*" (Cook, 2001). Sillitto stated, "*physically, a system of system looks just like a (big, spread-out) system with the following characteristics:*

- *Managerial and operational independence of the elements*
- *The elements have purpose and viability independent of the system of systems*
- *procured asynchronously, different budgets*
- *Not necessarily specified to be compatible*
- *May be competing against each other for budget and resources*
- *Emergent properties created by action at a distance through sharing information,*
- *system of systems is continually operating (or ready to operate),*
- *Key attributes are agility and dependability,*

System projects must be integrated into the "live" system of systems during operations" (Sillitto, 2008).

26.6.2 The reality

The reality is that there is recognition that systems exist within a hierarchy of systems in the context of adjacent systems and one person's system is another person's subsystem. The characteristics of systems of systems described above are the characteristics of systems in Layer 3 of the HKMF. For example, Sillitto's description would apply to the Allied convoys in the North Atlantic Ocean in World War II. Optimizing those convoys was a problem that was solved using Operations Research[160].

Other uses of the term "system of systems" describe an exploded view of a system containing several layers in the hierarchy of systems in a single drawing, where one person's subsystem is another person's system

The problems being addressed are those that Operations Research was set up to address in the 1940s and the tools and techniques exist and have existed for the last 50 years. Tools for systems engineering in the 1950s and 1960s were (Au and Stelson, 1969; Chestnut, 1965):

- Probability
- Single thread – system logic
- Queuing theory
- Game theory
- Linear programming
- Group dynamics
- Simulation
- Information theory

These tools were mainly used in the early stages of systems engineering. Since these early stages of systems engineering had been ignored in the standards, and the text books followed the standards, over time tools for systems engineering devolved to (Eisner, 1988; Jenkins, 2005):

- PowerPoint
- Databases (e.g. DOORS and CORE)
- Word processors
- Spreadsheets
- Drawing tools (e.g. Visio)
- Etc.

The myth arose when systems engineers educated and practicing in the HKMF Layer 2 US DOD systems engineering paradigm (DOD 5000.2-R, 2002) lacking the tools of the 1950s and 1960s attempted to tackle HKMF Layer 3 problems. Complexity is in the eye of the beholder (Jackson and Keys, 1984);

[160] Operational Analysis in the UK.

yes, it is a new class of problem to the HKMF Layer 2 systems engineers, and no, current operations research tools and techniques that deal with "systems of systems" might need to be modified, but new tools do not need to be developed; such tools do indeed exist and have existed for more than 50 years.

26.7 Myth 6: Changing requirements are a cause of project failure so get the requirements up front

26.7.1 The myth

The myth arose from:

1. the failure to capture the entire problem/need and create the full set of matching specifications for the solution system in the early phases of systems engineering, and
2. overlooking the fact that requirements change continuously and failure to manage that change is the cause of project failure.

There is thus confusion between the original uncaptured requirements and those requirements that arise due to changes.

26.7.2 The reality

The reality is that requirements may be categorized by:

1. those that exist at the time the solution system is specified and
2. those that come into existence while the system is being realized.

Definitely elicit and elucidate the known requirements in the early stage systems engineering activities. However, plan to use available tools and techniques such as configuration management, stage gates and engineering change processes to manage changes in requirements.

26.8 Myth 7: The single systems engineering process

This myth and corresponding reality was discussed in section 25.2 which explains the conflicting and contradictory information in the various versions of the systems engineering processes by viewing them as different unique subsets of the meta- systems engineering process appropriate to their situation. Consequently, there is single systems engineering process, which is different for every system development project and is really the SDLC containing a number of iterations of the problem solving process.

26.9 Summary

This Chapter has stated that systems engineering is currently a discipline characterized by debates based on subjective opinions, with participants talking past each other, a lack of listening and a number of myths. The Chapter discussed seven myths of systems engineering and showed the nature of each myth and the reality, and explained how and why each myth arose.

26.10 Conclusion

This Chapter has documented some findings about the current state of systems engineering. These findings are based on research into the history and practice of systems engineering. The findings of the research should provide food for thought and assist educators to improve the teaching of systems engineering.

2011

27 Seven principles for systems engineered solution systems

Systems engineering is presently demonstrating the characteristics of being in the emerging stages of a discipline. A discipline generally matures when an overriding axiom is presented and accepted by the majority of practitioners. This Chapter presents one such high level underpinning axiom for systems engineering that has the potential to unite the disparate camps within systems engineering and enable the practice of systems engineering in all application domains to achieve successes similar to those it achieved in the NASA environment in the 1960's and 1970's. The axiom does this by focusing on the solution system rather than on systems engineering.

27.1 The camps in systems engineering

Although systems engineering has been in existence since the 1940's, it is still demonstrating the characteristics of being in the emerging stage of a discipline. The characteristics of a discipline in this stage include application successes and failures as well as debates based on subjective opinions by participants in different camps talking past each other and a general lack of listening. The current somewhat overlapping camps in systems engineering are described in Section 29.2[161].

27.2 Towards unification

One reason for these debates and the camps is that systems engineers have different opinions on the nature of systems engineering. This is because:

[161] Section 29.2 contains an expanded version of the text originally published in the paper upon which this Chapter is based, so the topic is discussed there to avoid duplication.

- of the devolution of systems engineering over the past 60 years from a holistic paradigm to a stove-piped paradigm;
- systems engineering is so broad that systems engineers working on one part of the "systems engineering process" face different problems and perform different activities to those working in another part;
- they work in different application domains; and
- They often can't see the big picture perspective of systems engineering

These different opinions of systems engineering can be represented by the situation portrayed in the parable about the blind men feeling different parts of an elephant and deducing different animals. The parable[162] which is told in verse in an Indian setting (Saxe, 1873) pages 77 and 78) as quoted by (Yen, 2008) ends with the following stanzas.

"And so these men of Indostan
Disputed loud and long,
Each in his own opinion
Exceeding stiff and strong,
Though each was partly in the right,
And all were in the wrong!
MORAL.
So oft in theologic wars,
The disputants, I ween,
Rail on in utter ignorance
Of what each other mean,
And prate about an Elephant
Not one of them has seen!"

Systems engineering will not and cannot re-emerge as a unified discipline until the majority of the practitioners understand the situation, realise the big picture, and progress past these debates. From the temporal perspective, a discipline matures when one or more underpinning axioms have been hypothesized, presented to, and eventually accepted by the community. This acceptance occurs either when the axiom can represent the views of all or nearly all of participants in the debates, or, when one view becomes the dominant paradigm over the course of time. This Chapter presents one such high level underpinning axiom for systems engineering; namely "seven principles for systems engineered solution systems". While the axiom applies to systems engineering, the principles apply to the finished product irrespective of the systems engineering camp producing the solution system.

[162] The parable is said to have originated in China sometime during the Han dynasty (202 B.C. – 220 A.D.).

27.3 Seven principles for systems engineered solution systems

This Chapter presents one such high level underpinning axiom for systems engineering. While the axiom applies to systems engineering, the principles apply to the finished product irrespective of the systems engineering camp producing the solution system. This approach avoids presenting principles specific to particular camps which may not be accepted by systems engineers in the other camps.

Hitchins attributed the success of systems engineering in the NASA environment in the 1960's and 1970's to a set of eight principles (Hitchins, 2007) page 85). However, those principles applied in an environment where NASA's mission needs did not change very much during the SDLC for each mission. Today's systems on the other hand, tend to be developed and exist in an environment where the needs change, sometimes even before the solution system is delivered. This Chapter now introduces a set of principles for today's environment so that systems engineers working in different domains using various tools, techniques and methodologies, can meet the objective of systems engineering by applying the following set of principles to the solution system they are realizing:

1. There shall be a clear, singular objective or goal.
2. There shall be a CONOPS from start to finish of the mission describing the normal and contingency mission functions as well as the normal and contingency support functions performed by the solution system that remedies the problem.
3. The solution system shall be designed to perform the complete set of remedial mission and support functions for the operational life of the system.
4. The solution system design may be partitioned into complementary, interacting subsystems.
5. Each subsystem is a system in its own right, and shall have its own clear CONOPS, derived from, and compatible with, the CONOPS for the whole.
6. Each subsystem may be developed independently and in parallel with the other subsystems provided that fit, form, function and interfaces are maintained throughout.
7. Upon successful integration of the subsystems, the whole solution system shall be subject to appropriate tests and trials, real and simulated, that expose it to extremes of environment and hazards such as might be experienced during the mission.

Consider each of these principles.

27.3.1 There shall be a clear, singular objective or goal

Principle 1: There shall be a clear, singular objective or goal.

The task of the systems engineer shall have a clear singular objective goal. In the concept definition stage of a systems acquisition, this goal may be to identify the underlying problem or root cause of a situation, and to conceive one or more potential solutions. In the later phases of the solution SDLC[163] the goal is generally to realize a solution system that remedies the problem. For example, in the 1960's the NASA goal was to put a man on the Moon and return him safely to earth by the end of the decade. Similarly in the LuZ SEGS-1 system (Section 18.9) the goal was to provide a system that would convert solar energy to electrical power.

27.3.2 There shall be a clear CONOPS from start to finish of the mission …

Principle 2: There shall be a CONOPS from start to finish of the mission describing the normal and contingency mission functions as well as the normal and contingency support functions performed by the solution system that remedies the problem.

The CONOPS documents, or is a repository of, the information pertaining to the normal and contingency mission and support[164] performance of the overall solution system. One way of grouping the complete set of functions performed by any system is into the following classes:

- **Mission:** the functions which the system is designed to perform to provide a solution to the problem as and when required.
- **Support:** the functions the system needs to perform in order to be able to perform the mission as and when required. Support functions can further be grouped into (Hitchins, 2007) pages 128-129):
    - **Resource management functions** – the functions that acquire, store, distribute, convert and discard excess resources that are utilized in performing the mission.
    - **Viability management functions** – the functions that maintain and contribute to the survival of the system in storage, standby and in operation performing the mission.

---

[163] The notion that the solution system is generally a technological system that needs to be developed, and hence the name system development lifecycle, seems to be DOD inspired. Essentially, there need not be any (technological) development; instead, solution systems can be synthesized by bringing together existing systems to create a new unitary whole.

[164] The repeated use of "*normal and contingency mission and support*" is to emphasize the holistic approach.

Part of the CONOPS considers the consequences of failures of parts of the system to perform their mission and support functions and the contingency functions to be invoked in the event of these failures. The contingency functions may be in the process and consist of activities that will attempt to prevent the failure, or may be in the solution system in the form of viability functions.

The CONOPS is the foundation document[165] for both the solution system and the rest of the system realization activities since the remaining work in the SDLC realizes the solution system by converting the mission and support functions described in the CONOPS into a real system. Application of this principle leads to a holistic system development approach ensuring that all pertinent mission and support functions, such as operational availability, logistics, human operations, threat neutralizations, etc. are included in the system up-front in an integrated holistic manner and not as a bolt-on after the fact. A clear vision of the solution system anticipates, and consequently prevents, subsequent activities that try to clarify the original customer's problem represented by a set of poor requirements. The consequences of not having a CONOPS are shown in Figure 27-1[166]. I found this drawing in 1970 and it was old then. It has evolved somewhat in the intervening 40 years but the message it contains has not changed.

As an example of the benefits of a CONOPS, the (top level) CONOPS for the command and control system in Luz SEGS-1 (Section 18.9) was simply to generate electrical power using solar energy as the fuel. This mission was to be accomplished by deploying a field of parabolic trough reflector mirrors each morning, following the movement of the sun during the day to keep the mirrors focused on the sun and then stowing the mirrors in the evening when the sun set below the horizon. The support functions were to keep the mirrors clean, and to repair and maintain the elements of the system.

The CONOPS can also serve as a model of the solution and be incorporated in a simulation to allow various stakeholders to gain a better understanding of the problem space and determine if, and how well, the conceptual system being modelled could remedy the problem should that conceptual solution system be realized.

A CONOPS also facilitates elucidating requirements since the stakeholders, by agreeing on the CONOPS agree on the mission and support functions that the solution system will perform. The benefit of a shared stakeholder vision is discussed with reference to the Multiple-Satellite Operations Con-

[165] The word 'document' is used herein to represent information, not a necessarily a paper document.

[166] While it is often used to depict the "systems engineering process", it really shows a lack of communications or common vision of what the customer wants by the stakeholders.

trol Center (MSOCC) replacement switch case study in eliciting and elucidating requirements (Kasser and Mirchandani, 2005). The CONOPS can even minimize the number of requirements needed for certain types of systems such as LuZ's SEGS-1 because the CONOPS communicates the functionality and performance that must be developed and proven.

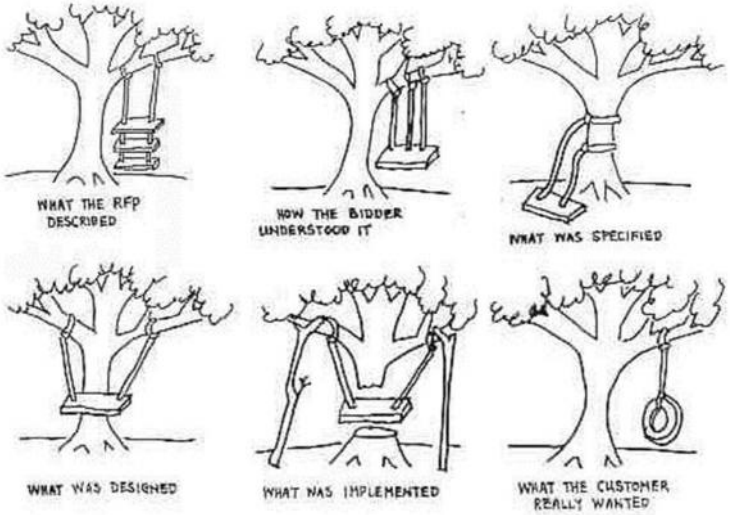


Figure 27-1 The consequences of not having a CONOPS

27.3.3 The solution system design …

Principle 3: The solution system shall be designed to perform the complete set of remedial mission and support functions for the operational life of the system.

The application of this principle produces a solution system that performs the mission and support functions described in the CONOPS over the complete lifecycle of the solution system. The solution system does not have to be technological or even a new acquisition. The solution system lies somewhere along a continuum that stretches from 'fully automatic technological' to 'manual with no technology'; and may be a modification of an existing system, a change to an existing process, tactics, doctrine, policy, or training or some combination. However, when applied to technological solution systems, this principle helps to ensure that the effects of component obsolescence, Diminishing Manufacturing Sources and Material Shortages (DMSMS), logistics, reliability, maintainability, the human element and other pertinent factors currently considered somewhat independently are considered interdependently in a holistic interdisciplinary manner from conception. Further, if the solution system is designed to perform in a hazardous or

threatening context, then the solution system shall incorporate support functions to counter threats and to manage risks.

27.3.3.1 Coping with change is a design criterion

This principle takes into account changes in/to the need/problem at any point in the SDLC. For example, in NASA's Apollo program, the need (and hence the requirements) did not change during the SDLC, and the operational life of each iteration of the manned element of the system was short; measurable in days. Each Apollo Lunar Surface Experiments Package (ALSEP)[167] however had a much longer life span.

Other early successful projects such as the transcontinental US television microwave relay system (Hall, 1962) were also not subject to changing needs. However, today's solution system creation and realization process must be able to cope with changes in the needs before the solution system is delivered, and the solution system itself needs be realized in such a manner that upgrades reflecting changing needs during its operational phase can be incorporated without major perturbations.

27.3.3.2 Cost is not an initial design criterion

According to this principle, the cost-effectiveness of the solution system is not a design criterion at least as far as the prototype or initial version is concerned. Once the prototype is shown to meet the needs, then costs may become an issue if the prototype is not affordable. Henry Ford wrote "*our policy is to reduce the price, extend the operations and improve the article. You will notice that the reduction of price comes first. We have never considered costs as fixed. Therefore we first reduce the price to a point where we believe more sales will result. Then we go ahead and try to make the price. We do not bother about the costs. The new price forces the costs down. The more usual way is to take the costs and then determine the price, and although that method may be scientific in the narrow sense, it is not scientific in the broad sense because what earthly use is it to know the cost if it tells you that you cannot manufacture at a price at which the article can be sold?*" (Ford and Crowther, 1922) page 146). It is a question of perspective and asking the right question. The usual non-holistic thinking question was "what does it cost to produce X?" From the Continuum perspective, the alternative (out-of-the-box) question was "how can X be produced for $Y?"

NASA's Apollo programme was more concerned with doing the job (meeting the goal of placing a man on the moon by the end of the 1960's) rather than doing it efficiently – money was not an issue in the initial design

[167] A set of scientific instruments deployed at the landing site of Apollo 12 to 17 designed to operate for a year. Each ALSEP contained the same central station and a slightly different set of scientific instruments.

phase. When the systems engineer designs each of the solution system options, cost and schedule must not be an issue. Cost and schedule considerations may be used as selection criteria for choosing the desired solution system option after the solution system options have been designed. In addition, systems engineers should be involved in any adjustments to the scope of the solution system realization project to fit the constraints of cost and schedule.

27.3.4 The solution system design partitioning …

Principle 4: The solution system design may be partitioned into complementary, interacting subsystems.

The solution that remedies the problem is the sum of the mission and support functions performed by the solution system and the functions performed in the realization process (Hall, 1989). Consider:

- Partitioning the product or solution system.
- Partitioning the production process.

27.3.4.1 Partitioning the product or solution system

The systems engineers design the solution system so that the desired functionality emerges from the complete design. For example, the performance of NASA's Apollo Moon Mission was emergent, coming as it did from the cooperation and coordination of the Saturn V launcher, the command module, the mission crew, the lunar excursion module, the telecommunications subsystem, mission control subsystem, etc. Performance is emergent because these various subsystems of the whole are of dissimilar nature, yet cooperate and coordinate their different functions and actions. So, you cannot point to any one subsystem and say – *'performance was down to that one'*. All parts contributed, all cooperated and coordinated their actions.

27.3.4.2 Partitioning the production process

The systems engineers also architect the activities that will constitute the realization process as interdependent streams of work between milestones (Chapter 19.)

27.3.5 Each subsystem is a system in its own right …

Principle 5: Each subsystem is a system in its own right, and shall have its own clear CONOPS, derived from, and compatible with, the CONOPS for the whole.

This principle reflects the observation that systems exist within containing systems and incorporates the structural hierarchical perspective. The principle has often been stated as "*one person's system is another person's subsystem*". Hierarchies are fundamental to nature.

As an example consider an allied naval convoy crossing the North Atlantic Ocean in 1942. The convoy is a system. Each ship in the convoy can be considered as both a subsystem of the convoy, or as a system[168]. There was a CONOPS for the convoy. There were separate CONOPS for the naval escort ships and the merchant vessels describing the actions and interactions of these subsystems of the convoy in various scenarios.

27.3.6 Each subsystem may be developed independently and in parallel …

Principle 6: Each subsystem may be developed independently and in parallel with the other subsystems provided that fit, form, function and interfaces are maintained throughout.

Each subsystem, being a system, needs its own systems engineers who conceive, design and develop their [sub]system as an interacting part of the containing system. These [sub]system systems engineers face in two directions – upwards and outwards into the containing system, to ensure ongoing compatibility with the containing system and its CONOPS, including all of the other interacting subsystems at the same level in the hierarchy; and downwards, into the intra-acting sub-subsystems within their own [sub]system. The downward task of developing the subsystems (function) can be considered as engineering when the focus is on the [sub]system as an independent entity.

During the realization phases of the SDLC (activities that take place in columns C … F of the HKMF) when the subsystems are being developed in parallel, the systems engineering activities are those that focus on the subsystem as a part of the complete system and ensure that fit, form and interfaces are maintained. If the SDLC takes a long time, the effect of changes in the need on the subsystem realization has to be taken into account. Experience has shown that subsystem designs and development may be subject to "creep". Consequently, it is necessary to have budgets for the whole system, as well as budgets for each of the subsystems — for instance the weight budget was important to Apollo, as was a failure rate budget. It would not have done for the failure rate for one subsystem – say the capsule – to go off the scale! Technical budgets have become known as 'Technical Performance Measures'. This is what is meant, in part, by conceiving, de-

[168] Alternatively, the naval ships could be one subsystem and the merchant marine ships a second subsystem of the convoy. Each ship is then a subsystem within the naval or civilian subsystem of the convoy. If there are ships from the navies of more than one allied country in the convoy, then the ships of each country could constitute a subsystem within the naval subsystem..

signing and developing the subsystem independently but within the context of the whole and the other interacting subsystems.

27.3.7 Upon successful integration of the subsystems …

Principle 7: Upon successful integration of the subsystems, the whole solution system shall be subject to appropriate tests and trials, real and simulated, that expose it to extremes of environment and hazards such as might be experienced during the mission.

This principle minimizes situations in which solution systems are delivered that are not fit for purpose and do not provide a solution in the intended environment.

The consequences of not implementing this principle can be seen in the increasing stovepiping of the processes in the SDLC and expansion of the various disciplines. For example, consider the expansion of T&E discussed in Chapter 9. The questions raised by the T&E personnel should be addressed when developing the CONOPS.

27.4 Discussion

Poor systems engineering has been blamed for system acquisition failures[169] according to many sources including (Wynne, 2004). An objective view might suggest that budget and time overruns smack of either poor estimating of cost and schedules or understating the real estimates for reasons that appeared valid at the time. However, in all fairness, poor early stage systems engineering does seem to have been a contributor to some of those failures resulting from producing solutions systems that do not remedy the need when deployed. Attempts to mitigate the effects of poor early stage systems engineering in the early stages of the system have resulted in system development becoming increasingly technologically focused, excessively complicated and stove-piped into independent streams of activities including:

- Systems Engineering
- Project Management
- Lifecycle Costing or Total Ownership Cost
- Performance Based Logistics
- Integrated Logistics Support
- Maintenance Management
- Supply Chain Management
- Technical Training Management

[169] Defined herein as cost and schedule overruns, cancellations and delivered systems that are not fit for purpose.

- Technical Data Management
- Configuration Management
- Risk Management
- Independent Verification and Validation
- Human Systems Integration

These are but some examples of the independent streams of activities in the various specialties in the SDLC. Not only is this stove-piping against the holistic concept of systems engineering, stove-piping produces overlapping activities, confusion, and unnecessary expense and also provides a breeding ground for turf wars in organizations. The documentation overhead is increasingly becoming expensive and documents that should be interdependent are independent being produced because of legislation rather than as a result of actual need. Today's DOD systems engineering paradigm has added so many bolt-ons to compensate for having removed the front end of systems engineering that it has become expensive and unworkable (Costello, 1988). Reversion to the original holistic *weltanschauung* (world view or paradigm) is long overdue since in the 20 years since the Costello Report was published, the situation has worsened.

27.5 Summary

Systems engineering is presently demonstrating the characteristics of being in the emerging stages of a discipline. A discipline generally matures when an overriding axiom is presented and accepted by the majority of practitioners. This Chapter presented one such underpinning axiom for systems engineering. The principles within the axiom apply to the solution system, production of which is the common goal of all the camps within systems engineering. As a consequence, the axiom has the potential to unite the disparate camps within systems engineering by allowing them to agree on the principles applying to the solution system which will then enable the practice of systems engineering to repeat the successes it achieved in the NASA environment in the 1960's and 1970's in all current and future application domains.

27.6 Conclusion

The principles presented in this Chapter apply to the solution system being systems engineered rather than to systems engineering. As such the axiom has the potential to unite the disparate camps within systems engineering by allowing them to agree on the principles. Applying these principles to the solution system will then enable the practice of systems engineering to repeat the successes it achieved in the NASA environment in the 1960's and 1970's in all current and future application domains.

2012

28 Getting the right requirements right

Research has shown that there is an on-going consensus that:

1. good requirements are critical to the success of a project,
2. the current requirements paradigm produces poorly written requirements and
3. ways of producing better requirements have been around for more than 20 years.

So, instead of producing yet another opinion on how to write better requirements, this Chapter begins by posing the following question:

why do systems and software engineers continue to produce poor requirements when ways to write good requirements have been documented in conference papers and textbooks?

The Chapter then documents findings from research into the problem via the holistic thinking perspectives and hypothesises that there are two requirements paradigms; the original A paradigm and the current B paradigm which is inherently flawed. The Chapter then dissolves the problem of poor requirements by applying technology to reduce the need for most of the requirements written today.

28.1 The perennial problem of poor requirements

Requirements drive the work in the SDLC as shown in Figure 17-2. Research has shown that there is an on-going consensus that (Kasser and Schermerhorn, 1994a; Jacobs, 1999; Carson, 2001; Hooks, 1994; DOD, 2011; Goldsmith, 2004; Wheatcraft, 2011; Lee and Park, 2004; Jorgensen, 1998):

1. good (well-written) requirements are critical to the success of a project,
2. there is no metric for the “goodness” of a requirement,

3. the current requirements paradigm produces poorly written requirements and
4. ways of producing better requirements have been around for more than 20 years.

So, instead of producing yet another opinion on how to write better requirements, this Chapter begins by posing the following question: why do systems and software engineers continue to produce poor requirements when ways to write good requirements have been documented in conference papers and text books? (Jorgensen, 1998; Lee and Park, 2004; Alexander and Stevens, 2002)

28.2 The perspectives perimeter

This section on the HTPs (Kasser, 2013) was grafted into this Chapter since the papers describing the systems and holistic thinking perspectives (HTP) alluded to in the preface to the second edition were not included in this volume. This section introduces a set of viewpoints on the perspectives perimeter which can be used to provide anchor points for thinking and communicating in a systemic and systematic manner. These viewpoints go beyond combining analysis (internal views) and systems thinking (external views) by adding quantitative and progressive (temporal, generic and continuum) viewpoints. The nine HTP anchor points shown in Figure 28-1 are:

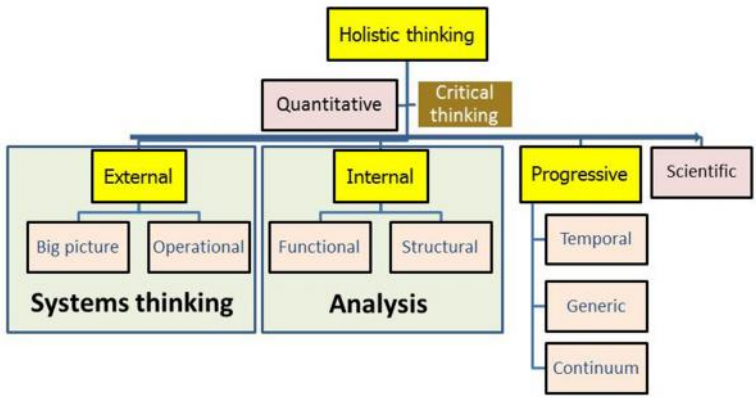


Figure 28-1 Holistic thinking perspectives (structural view)

- **External perspectives:** The External perspectives are:
  1. ***Big Picture:*** the context for the system.
  2. ***Operational:*** what the system does.
- **Internal perspectives:** The Internal perspectives are:

3. ***Functional:*** what the system does and how it does it.
4. **Structural:** how it is constructed and organised.

- **Progressive perspectives:** The Functional and Structural perspectives provide internal views; the Big Picture and Operational perspective provide external views. The progressive perspectives are where holistic thinking begins to go beyond analysis and systems thinking and are orthogonal to the internal and external perspectives as shown in Figure 28-2. The progressive perspectives are:

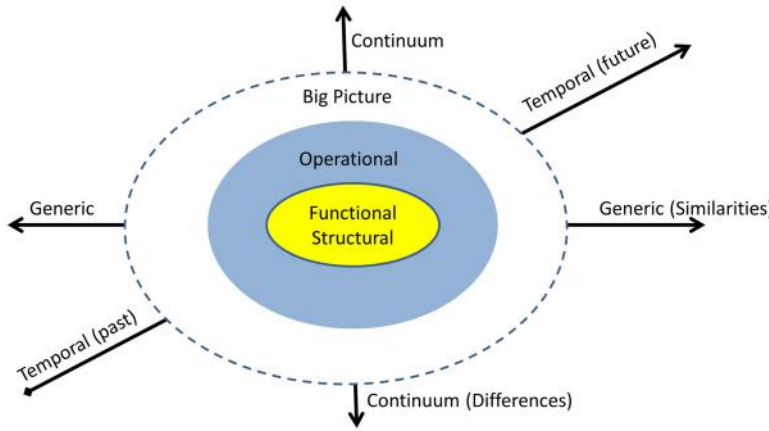


Figure 28-2 Holistic thinking perspectives

5. ***Generic***: where the system is perceived as an instance of a class of similar systems.
6. ***Continuum:*** where the system is perceived as but one of many alternatives.
7. ***Temporal:*** which considers the past, present and future of the system.

- **Other perspectives:** The other perspectives are:

8. ***Quantitative:*** the numeric and other quantitative information associated with the system.
9. ***Scientific:*** the hypothesis or guess about the issues

28.3 Situational analysis

This section analyses the situation from the HTPs to develop an answer to the question "why do systems and software engineers continue to produce poor requirements when ways to write good requirements have been documented in conference papers and textbooks?" posed above.

28.3.1 Big picture perspective

There seem to be a number of reasons for systems and software engineers to continue to produce poor requirements when ways to write good requirements have been documented in conference papers and text books, including (in no particular order):

- Lack of time to write the requirements due to schedule constraints, which results in poorly drafted and incomplete requirements.
- Failure of the stakeholders to articulate the requirements, which results in incomplete and sometimes results in incorrect requirements.
- Lack of training for writing good requirements sometimes due to budget issues, which results in poorly written requirements.
- Fundamental lack of understanding of the need for, and the purpose served by, requirements, by management, which results in lack of sufficient time for the requirements elicitation and elucidation process.
- Lack of implementation and solution domain knowledge in the systems and software engineers eliciting and elucidating the requirements, which tends to result in incomplete and sometimes unachievable requirements.
- Lack of functionality in commercial requirements tools that can call attention to requirements that are poorly written.

These reasons can be aggregated into two issues:

1. Production of poorly written requirements.
2. Lack of ways of ensuring completeness of the requirements.

28.3.2 Operational perspective

Requirements are written in the context of the system acquisition and design process which takes place in the front-end of the SDLC[170]. Since various depictions of this process exist (Chapter 25), this Chapter uses the representation of the system design process shown in Figure 28-3[171] (Bahill and Dean, 1997) as a typical example. This version of the system design process begins with a customer request, which is then analysed, and a problem statement developed. Requirements, behavioural scenarios and models are then produced in parallel to develop a number of conceptual solution systems. After a feasible conceptual solution has been selected, the process to realize the

[170] The notion that the solution system is generally a technological system that needs to be developed, and hence the name system development lifecycle, seems to be US Department of Defense inspired. Essentially, there need not be any (technological) development; instead, solution systems can be synthesized by bringing together existing systems to create a new unitary whole.

[171] The figure seems to use the terms “solution” and design” interchangeably.

solution system (for the following phases of the SDLC) is designed (Chapter 19) and the SDLC continues.

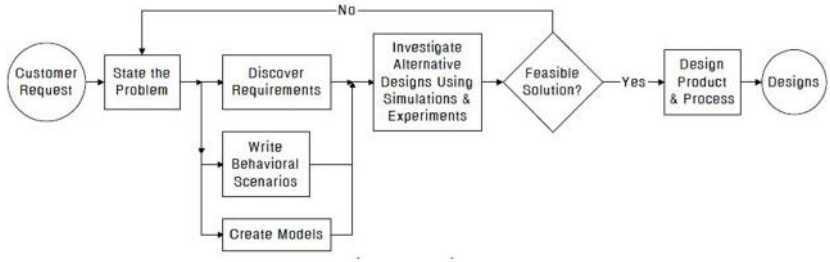


Figure 28-3 System design process (Bahill and Dean, 1997)

28.3.2.1 Overlapping streams of work

One of the consequences of the focus on the functional and performance aspects of the solution system and the neglect of the non-functional attributes has been an increase in the complexity and cost of the acquisition process. This is due partly to the growth of the activities performed in T&E, Integrated Logistics Support and Configuration Management to ensure that the systems acquired meet the needs of the user when fielded (irrespective of quality and completeness of the requirements). These activities now form parallel overlapping streams of work within the SDLC, generate nugatory work producing legally required documents that make little if any contribution to the success of the project while escalating the costs. Consider the following three examples of the overlaps:

- SEMP and TEMP.
- Logistics Support Analysis (LSA).
- Configuration Management.

28.3.2.1.1 SEMP and TEMP

The contents of a TEMP overlap the contents of a SEMP (Florida, 2006).

28.3.2.1.2 Logistics Support Analysis (LSA)

Logistics Support Analysis (LSA) is defined as an activity within Integrated Logistics Support which generates a Logistics Support Analysis Record. The activity is defined as "*the iterative process of identifying support requirements for a new system, especially in the early stages of system design"*. The main goals of LSA are to ensure that the system will perform as intended and to influence the design for supportability and affordability. LSA, performed as integral part of system design (up front):

- Produces supportability requirements as an integral part of system requirements and design.
- Defines support requirements that are optimally related to the design and to each other.

- Defines the required support during the operation phase of the system.

28.3.2.1.3 Configuration Management

Configuration Management is defined as "a field of management that focuses on establishing and maintaining consistency of a system's or product's performance and its functional and physical attributes with its requirements, design, and operational information throughout its life" (MIL-HDBK-61A, 2001). There are two types of configuration audits within configuration management:

- **Functional configuration audits–** which ensure[172] that functional and performance attributes of a configuration item are achieved, and
- **Physical configuration audits** - which ensure that a configuration item is installed in accordance with the requirements of its detailed design documentation.

Configuration audits can occur either at delivery or at the moment of effecting a change. These audits are commonly known as verification and validation or testing in the systems engineering community.

28.3.3 Functional perspective

The functions or activities performed to discover requirements within Figure 28-3 are elaborated in Figure 28-4. This process is complicated containing activities to:

1. determine if requirements are feasible,
2. identify risks associated with requirements, and
3. detect correct and contradicting requirements in the step that asks why each requirement is needed.

However, the process cannot determine if the requirements are complete. As a result there is no way to show that the requirements express a complete solution that will meet the need until the solution system is actually fielded and put into service at the end of the SDLC, sometimes after many years and the expenditure of lots of money. As mentioned above, variations of this process exist. For example Jorgensen provides a variation on the process calling out the need for an evaluation of the operations concepts of a system before writing requirements (Jorgensen, 1998).

The front end of the Australian SDLC is different to the example shown in Figure 28-3. There, an OCD is created and the functional and performance requirements for the system are based on the OCD (Gabb, et al., 2001).

[172] Ensure does not necessarily mean do, it also means making sure something is done, such as in Quality Assurance.

Gabb stated that an OCD may include identification and discussion of the following:

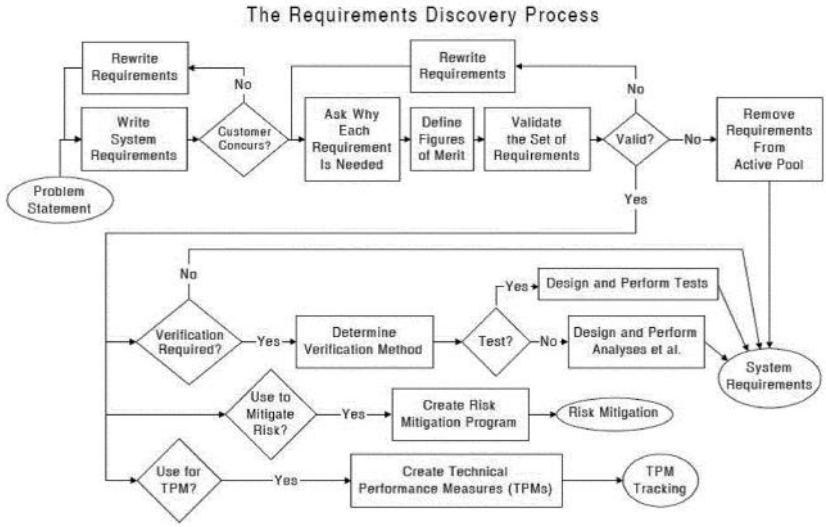


Figure 28-4 The requirements discovery process (Bahill and Dean, 1997)

- Why the system is needed and an overview of the system itself.
- The full system lifecycle from deployment through disposal.
- Different aspects of system use including operations, maintenance, support and disposal.
- The different classes of user, including operators, maintainers, supporters, and their skills and limitations.
- Other important stakeholders in the system.
- The environments in which the system is used and supported.
- The boundaries of the system and its interfaces and relationships with other systems and its environment.
- When the system will be used, and under what circumstances.
- How and how well the needed capability is currently being met (typically by existing systems).
- How the system will be used, including operations, maintenance and support.

The OCD is presented at an Operations Concept Review (OCR) and analysed in the subsequent phase of the SDLC and then used to create the functional and performance requirement documents.

28.3.4 Temporal perspective

From this perspective, one can examine how the need for requirements

evolved and where requirements are produced and used in system acquisition and in the SDLC.

Research[173] shows that the early systems engineers of the 1950's and 1960's tended to focus on identifying the problem (Wymore, 1993) and finding an optimal solution (Hall, 1962; Goode and Machol, 1959). These early systems engineers were Types III, IV, and V, (Kasser, et al., 2009) while the systems engineers who came later tended to focus on processes (Type II)'s. Back in the "good old days" of systems engineering, Type III, IV and V systems engineers remedied the problem in the early stage systems engineering activities addressing the conceptual solution. They then produced the matched set of specifications for the implementation or realization of the solution, and moved on to the next contract, leaving the Type II's to continue realizing the solution. There then came a time when there was a lack of new projects and so many of the Type III, IV and V's were laid off and lost to the discipline. When the need for systems engineers picked up again, in general, only the Type II systems engineers were left and they took over systems engineering. They had seen a successful process for developing systems following the production of the matched set of specifications and so their focus was on the post-requirements phases of the SDLC. They wrote the standards used in systems engineering (MIL-STD-499, 1969; MIL-STD-499A, 1974; EIA 632, 1994; IEEE 1220, 1998) for other Type II systems engineers to follow. As a consequence, the critical early stage engineering activities addressing the problem and conceptual solution were left out of mainstream Type II systems engineering (Bruno and Mar, 1997; Fisher, 1996). The Standards in turn became the foundation for educating systems engineers.

28.3.5 Quantitative perspective

While there is a consensus that requirements are critical, and that requirements suffer from several types of defects (see structural perspective), there is no widely accepted metric for the goodness of requirements (Kasser, et al., 2006) nor does there seem to be a widely accepted baseline definition of a requirement. For example the IEEE definition of a requirement is (IEEE 610, 1990):

"(1) A condition or capability needed by a user to solve a problem or achieve an objective.

(2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.

A documented representation of a condition or capability as in (1) or (2)."

[173] The evolution of the role of systems engieering was discussed in section 23.1.

Yet variations of the definition continue to appear in the literature, including:

- Something that is wanted or needed, called for or demanded as being essential (Mason, et al., 1999).
- A statement which translates (or expresses) a need or constraints (technical, costs, times ...). (Fanmuy, 2004).
- Something obligatory or capabilities the system must satisfy (Powell and Buede, 2006).
- Kossmann et al. cite a number of definitions in the literature and also provide a useful overview of the state-of-the-art of requirements engineering based on a wide collection of publications from previous years (Kossmann, et al., 2007).

28.3.6 Structural perspective

A text-mode requirement should just be a simple sentence. Yet there are problems in the way requirement sentences are structured (Scott, et al., 2006). As discussed above, contemporary requirements management practice irrespective of the process used to generate the requirements is far from ideal, producing:

- **Vague and unverifiable requirements** – due to poor phrasing of the written text.
- **Incompletely articulated requirements** – due to a poor requirements elicitation process.
- **Incomplete requirements** – due to various factors including domain inexperience, and the lack of expertise in eliciting and writing requirements by technical staff.
- **Poor management of the effect of changing user needs during the time that the system is under construction** – due to lack of the understanding of the need for change management, and use of appropriate tools to do the function in an effective manner.

In conjunction with improving the writing of requirements, there also has been recognition that a requirement is more than just the imperative statement having additional properties (e.g. priority and traceability) (Alexander and Stevens, 2002; Hull, et al., 2002). The IEEE Computer Society Computing Curriculum - Software Engineering --- Public Draft 1 --- (July 17, 2003) Software Engineering Education Knowledge Software expands on the earlier IEEE 610 definition of a requirement as follows "*Requirements identify the purpose of a system and the contexts in which it will be used. Requirements act as the bridge between the real world needs of users, customers and other stakeholders affected by the system and the capabilities and opportunities afforded by software and computing technologies. The construction of requirements includes an analysis of the feasibility of the desired*

system, elicitation and analysis of stakeholders' needs, the creation of a precise description of what the system should and should not do along with any constraints on its operation and implementation, and the validation of this description or specification by the stakeholders. These requirements must then be managed to consistently evolve with the resulting system during its lifetime".

However, in practice, there is difficulty in adding these additional properties to the traditional requirement document or database and then managing them. This is because the current systems and software development paradigm generally divides the work in a project into three independent streams as shown in Figure 2-2. Thus requirements engineering tools contain information related to the Development and Test streams (the requirements) while the additional properties tend to be separated in several different tools, (e.g. Requirements Management, Project Management, Work Breakdown Structures, Configuration Control, and Cost Estimation, etc.).

28.3.7 Generic perspective

Text mode requirements are but one way to communicate information. Other ways of communicating all or part of the same information include models, simulations, photographs, schematics, drawings, and prototypes. Fanmuy clarifies the definition of a requirement statement by adding "*this statement is written in a language which can take the form of a natural language or a mathematical, arithmetic, geometrical or graphical expression*" (Fanmuy, 2004). Timing and state diagrams are often used in Requirements Documents. Thus the concept of stating user needs (under certain circumstances) via diagrams is already in use in systems engineering (Kasser, 2002b). Thus, from this perspective, requirements are but one of a number of communications tools. The focus should be on user needs, not on requirements. Van Gaasbeek and Martin quoted Dahlberg as stating, "*we don't perform system engineering to get requirements*" and, "*we perform system engineering to get systems that meet specific needs and expectations*" (Van Gaasbeek and Martin, 2001).

28.3.8 Continuum perspective

This perspective also looks at ranges and ambiguities. For example:

- The information in a requirements document can be looked at as both a solution and a problem. The matched set of specifications documents are a conceptual solution system that should remedy the problem. Thus, they document a solution as far as the customer is concerned but at the same time, they also document the problem faced by the designers who have to design the solution system.
- The word 'requirement' may have different meanings in different phases of the acquisition lifecycle. In the early stage systems engineering ac-

tivities in column A of the HKMF, 'requirement' and 'need' may also be used interchangeably, but have slightly different meanings in the worldviews of the customer and contractor. These differences in meanings show up in the IEEE definition of a requirement (IEEE 610, 1990).

28.3.9 Scientific perspective

There seem to be two requirements engineering paradigms.

A. The first paradigm begins with the systems engineering activities performed in column A of the HKMF shown in Figure 21-3 discussed in Section 21.9.
B. The second paradigm skips the column A activities and begins in column B.

28.4 The two requirements paradigms

Consider the two paradigms.

28.4.1 The A paradigm

The A paradigm beings with the systems engineering activities performed in column A in the HKMF. Research into the systems engineering literature found that successful projects such as the NASA Apollo program (Hitchins, 2007) and the LuZ SEGS-1 solar project (Chapter 22) were characterised by a common vision of the purpose and performance of the solution systems among the customers, users and developers; namely a paradigm that began in column A of the HKMF. Moreover, the common vision related to both the mission and support functions performed by the solution system. The research finding was supported outside the systems engineering literature by similar findings in the process improvement and Quality literature (Deming, 1993; Dolan, 2003). In addition, note that Business Process Reengineering creates and disseminates/communicates a 'to-be' model of the operation of the conceptual reengineered organisation before embarking on the change process.

Requirements are developed as an intermediate work product in the SDLC to provide formal communication between the stakeholders. Many informal situations do not need requirements when the vision is present. During the Luz system development there was a need for a position sensor to report on the angle between the focus of the mirror and the horizon (Chapter 22). Alternative approaches to identifying the position of the mirror examined were (a) a pulse counting based sensor, (b) an analogue sensor based on a potentiometer and (c) an absolute position optical shaft encoder sensor. The optical shaft encoder option was selected after due consideration. A search through the component catalogues quickly identified a COTS

Gray code[174] absolute position rotary encoder with an eight-bit parallel transistor-transistor-logic (TTL) interface. There was no need to write any requirements for the position sensor interface in this informal situation. The team had a concept of what the system was supposed to do; they had the (electronics) domain knowledge necessary to understand Gray code and the specifications on a TTL parallel interface, and got on with the interface design.

Since the A paradigm is characterized by a common vision of the purpose of the mission and support functions of solution systems among the customers, users and developers, the quality of the requirements tends to have little if any impact on the functionality of the solution system.

28.4.2 The B paradigm

Many systems and software engineers have been educated to consider the systems engineering activities in column B of the HKMF as the first phase of the systems engineering process. For example,

- requirements are one of the inputs to the 'systems engineering process' (Martin, 1997) page 95), (Eisner, 1997) page 9), (Wasson, 2006) page 60) and (DOD 5000.2-R, 2002), pages 83-84);
- in one postgraduate class at University of Maryland University College the instructor stated that systems engineering began for him when he received a requirements specification (Todaro, 1988).

While DOD 5000 does call out the 'analysis of possible alternatives' subset of activities performed in Phase A.2 of the HKMF (DOD 5000.2-R, 2002) pages 73-74), those activities are called out as part of the separate seemingly independent CAIV process. CAIV is a way of complicating just a part of the concept of designing budget tolerant systems using the cataract approach (Section 13) and takes place **before** the DOD 5000.2-R 'systems engineering process' begins.

28.5 Discussion

The B paradigm is inherently flawed. This is because even if systems and software engineers working in a paradigm that begins in HKMF column B could write perfectly good requirements and follow a process such as the one shown in Figure 28-5 (Guo, 2010), they still cannot determine if the requirements and associated information are correct and complete because

[174] A binary code, where two successive values differ in only one bit, originally designed to prevent spurious outputs during transitions from one state to another in electromechanical switches.

there is no reference for comparison to test for the completeness. Consequently, efforts expended on producing better requirements have not, and will not, alleviate the situation. The situation cannot be alleviated because the situation is akin to participating in Deming's red bead experiment, which demonstrates that errors caused by workers operating in a process are caused by the system rather than the fault of the workers (Deming, 1993) page 158). Recognition that the B requirements paradigm is inherently flawed is not a new observation. For example:

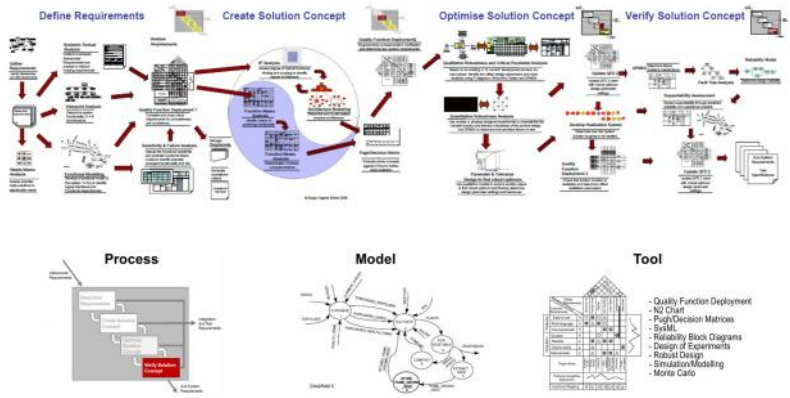


Figure 28-5 One example of the B paradigm (Guo, 2010)

- Sutcliffe et al. proposed reducing human error in producing requirements by analysing requirements using an approach of creating scenarios as threads of behaviour through a *use case*, and adopting an object-oriented approach (Sutcliffe, et al., 1999); namely they proposed a return to the A paradigm.
- Daniels et al. point out that standalone requirements make it difficult for people to understand the context and dependencies among the requirements, especially for large systems and suggest using use cases to define scenarios (Daniels, et al., 2005).
- One of the two underlying concepts of Model Based Systems Engineering (MBSE) is to develop a model of the system to allow various stakeholders to gain a better understanding of how well the conceptual system being modelled could remedy the problem, before starting to write the requirements. MBSE with its roots in the process camp of systems engineering and the B paradigm has discovered the CONOPS and is trying to return to the A paradigm.

28.6 Upgrading the A paradigm for the 21st century

In the second half of the 20th century the CONOPS, requirements specifications and associated information were stored in the form of separate docu-

ments containing text and graphics. In the latter years of the 20th century, information technology provided electronic storage capability in the form of databases. In the 21st century, the CONOPS and requirements can be linked via technology. Technology allows a multi-media approach to be used to communicate the vision of the solution system. This concept was described and prototyped as an Operations Concept Harbinger (OCH) which may be thought of as a multimedia OCD that also contains measures of effectiveness for each operational scenario (Kasser, et al., 2002). The OCH architecture consists of an underlying database and agents that act on the contents of the database to describe relationships and performance in various scenarios in the language of the viewer. Therefore, some designers could see the solution system execution expressed in UML, SysML, or IDEF0, and other stakeholders could see the solution as PowerPoint slides, videos, sound bites, etc. A prototype OCH was constructed in the SEEC at UniSA and used in a successful Force Level Systems Engineering application (Kasser, et al., 2002).

28.6.1 Relating the CONOPS to Requirements

The use of *use case*s within a CONOPS in an object-oriented approach describing 'properties of', and 'services' (functions) provided by, components, can often provide the same representation of user needs as that of "requirements" if each property consists of an attribute and a value (Section 16.7). Other non-functional attributes such as colour and weight associated with a component can also be shown in the property viewer. The object-oriented approach also provides for inheritance of attributes of various classes of components which helps to maximise the completeness of the information in the CONOPS. The relationship between the undesirable situation, CONOPS, functions and requirements in the A paradigm can be expressed as shown in Figure 28-6.

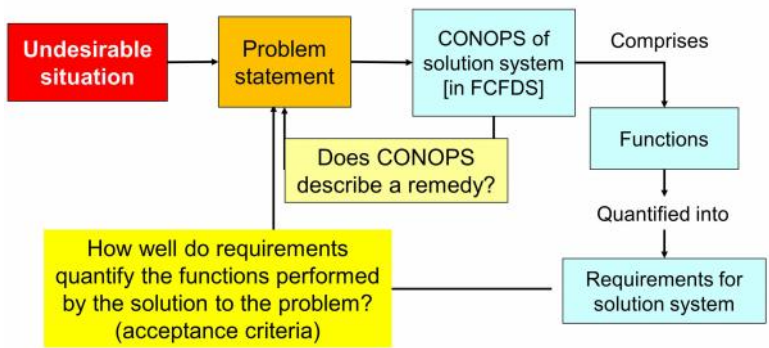


Figure 28-6 CONOPS, functions and requirements

The problem solving process in the early stages of systems engineering performed in column A of the HKMF are shown in Figure 28-7 (Hitchins,

2007): Figure 6.2)[175]. The solution options may be constructed in the form of an OCH or an information-technology based CONOPS often called a model rather than in a document-based format.

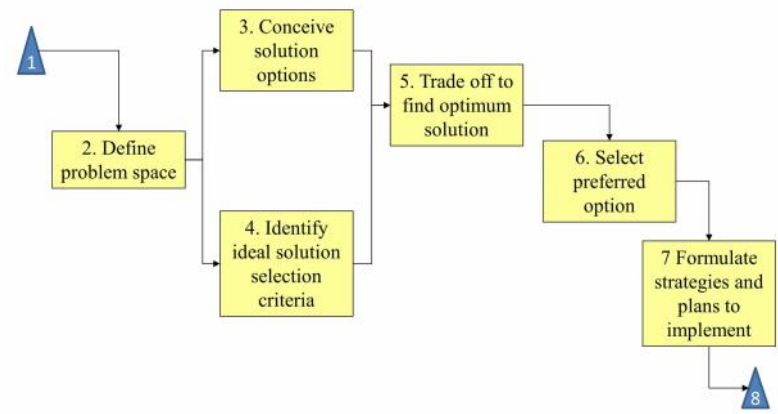


Figure 28-7 Hitchins problem solving process in the early stages of the SDLC

So, if the model or an object-oriented CONOPS (OOCONOPS) can represent the user's needs in a manner verifiable by all stakeholders, there is no need for writing many of the requirements that seem to be needed in the B paradigm. The process shown in Figure 28-4 can be simplified. For example, there is no need to question the need for requirements since the OOCONOPS contains that information. In fact, since the OOCONOPS can communicate the vision more thoroughly than do text mode requirements, the number of requirements can also be reduced[176]. Consequently, instead of producing a mixture or requirements and behavioural models, systems and software engineers need to produce a single OOCONOPS for each of the solution system options. These are discussed with the customer and other stakeholders to verify that the selected solution system depicted in the OOCONOPS meets the need and will remedy the problem.

The object-oriented paradigm encapsulates processes (functionality) as well as data into an object. The types of 'smart' functions that might be encapsulated within an OOCONOPS were discussed in Section 17.6. Inheritance is a major advantage of the object-oriented systems engineering paradigm since most new systems that are similar to, or can be considered as a class of, an existing system (Section 17.7).

---

[175] Which unlike Figure 28-3 is abstract enough not to describe the format of the solution options.

[176] Note, in some cases, some requirements may still be needed at the contractual boundaries between contractors and sub-contractors.

28.7 Discussion

The A paradigm is a return to the systems engineering paradigm of the good old days. In the A paradigm, the propensity for errors is much lower which provides an improvement in reliability and a reduction in the amount of work performed in the project (cost and schedule) as compared to the B paradigm which is inherently flawed. Getting the right requirements in the right way means minimizing the number of requirements and communicating the vision via an OOCONOPS. The remaining few requirements needed should be a printout from the OOCONOPS rather than being produced by a requirements analysis exercise. Today in some instances, requirements are bypassed when the contractor is asked to deliver according to a schematic or copy a prototype. Hence, using an OOCONOPS is not really a completely new way of requirements engineering. Requirements are a means not an end.

Figure 17-2 should be viewed in the context of the B paradigm, a better A paradigm version where the CONOPS drives the work is shown in Figure 28-8.

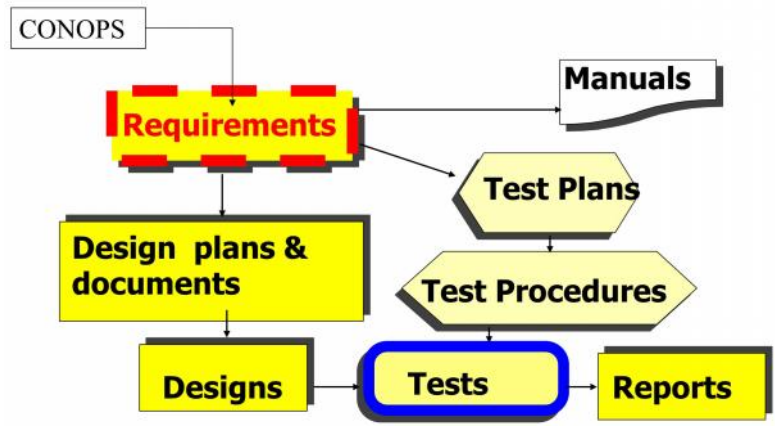


Figure 28-8 CONOPS drives work

28.8 Summary

The Chapter opened with the statement that research has shown that there is an on-going consensus that:

1. good requirements are critical to the success of a project,
2. the current requirements paradigm produces poorly written requirements and
3. ways of producing better requirements have been around for more than 20 years.

So, instead of producing yet another opinion how to write better requirements, this Chapter began by posing the following question: why do systems and software engineers continue to produce poor requirements when ways to write good requirements have been documented in conference papers and textbooks? The Chapter then documented findings from research into the problem via the systems thinking perspectives and hypothesised that there are two requirements paradigms; the original A paradigm and the current B paradigm which is inherently flawed. The Chapter then dissolved the problem of poor requirements by applying technology to reduce the need for most of the requirements written today.

28.9 Conclusion

The current type A requirements paradigm is inherently flawed and cannot be repaired. An OOCONOPS based paradigm can eliminate the need for many of the requirements necessary in the current requirements paradigm, increasing the probability of a successful project at lower cost. Accordingly, the OOCONOPS paradigm dissolves the problem of poor requirements by making most of the text-mode requirements used in the B paradigm irrelevant and unnecessary.

2012

29 Yes systems engineering, you are a discipline

> *"It ain't what you don't know that gets you into trouble. It's what you know for sure that just ain't so." – Mark Twain 1835-1910.*

Systems engineering is currently characterized by conflicting and contradictory opinions on its nature. This Chapter begins by describing the evolution of systems engineering in the NCOSE/INCOSE and the difficulty in defining and differentiating systems engineering as a discipline. The Chapter then identifies and discusses six different and somewhat contradictory camps or perspectives of systems engineering. After identifying the cause of the contradictions the Chapter suggests one way to reconcile the camps is to dissolve the problem to distinguish between the activity known as systems engineering and the role of the systems engineer with a return to the old pre-NCOSE systems engineering paradigm. The Chapter then continues by testing the hypothesis and shows that systems engineering is a discipline that can be differentiated from other disciplines. However, it is not a traditional engineering discipline.

29.1 Systems engineering in NCOSE/INCOSE

Systems engineers have had a problem, not only explaining what they do, to other people but also defining it amongst themselves since the early 1990's (Chapter 2). The baseline for this research however began at the 1994 symposium of the NCOSE, where presenter after presenter opened their presentation with a definition of systems engineering and each definition was different. However, when each presenter continued by describing the functions performed by systems engineers, they talked about planning, organizing, directing and controlling; the traditional functions of management (Fayol, 1949) page 8). When asked what systems engineers did, their answers were also different. These observations in 1994 triggered a research

program into the nature of systems engineering and its overlap with project management which began with an analysis of the activities or functions performed by systems engineers. The initial research showed that there seemed to be no unique body of knowledge to systems engineering and that all of the activities performed by systems engineers, apart from possibly requirements and interfaces, were also performed by other types of engineers (Chapter 2). Chapter 2 concluded with "*systems engineering is a discipline created to compensate for the lack of strategic technical knowledge and experience by middle and project managers in organizations functioning according to Taylor's "Principles of Scientific Management*". Subsequent research into the nature of systems engineering included a literature review of text books published between 1959 and 2009 starting with Goode and Machol (Goode and Machol, 1959) as well as the proceedings of all the international symposia of the INCOSE since 1991. Findings from this research determined that (Chapter 12 and 23):

- The role of the systems engineer in the workplace depends on the situation. This is because the role of the systems engineer has evolved over time so that it is different in practically every organisation and has various degrees of overlap with the roles of project managers and personnel in other disciplines.
- Definitions and descriptions of systems engineering comprise different interpretations of the broad raft of activities that systems engineers might undertake according to their role in the workplace.

This multichotomy exists because different people have chosen or perceived different meanings of the term 'systems engineering' for almost 60 years. Consider the following comment from 1960 "*Despite the difficulties of finding a universally accepted definition of systems engineering, it is fair to say that the systems engineer is the man who is generally responsible for the over-all planning, design, testing, and production of today's automatic and semi-automatic systems*" (Chapanis, 1960) page 357). Jenkins expanded that comment into twelve roles (activities performed by a person with the title systems engineer) of a systems engineer and seven of those roles overlapped the role of the project manager (activities performed by a person with the title project manager) (Jenkins, 1969) page 164). Since that time, systems engineering has evolved and some of the evolution in systems engineering can be seen in the very little overlap between Jenkin's twelve roles and the twelve systems engineering roles documented by Sheard in 1996 (Section 23.1).

29.2 The seven camps in systems engineering

An analysis of the different views of/opinions on/worldviews of systems engineering in the early years of the 21st century identified the following

somewhat overlapping camps: Lifecycle, Process, Problem, Discipline, Systems thinking and non-systems thinking, Domain and Enabler. Consider each of these camps:

29.2.1 Lifecycle camp

Some systems engineers seem to have an understanding of the A paradigm early stage systems engineering activities that take place in the concept definition stage of a solution system acquisition[177] (Chapter 28). The majority however live in the B paradigm and have no idea that the concept definition phase even exists, they don't understand what happens in that phase and they think that systems engineering in the acquisition domain begins with the requirements analysis phase. The early stage campers tend to be the old timers; while the others tend to be those systems engineers educated in the last 20-30 years in the B paradigm based on the US DOD where the whole set of activities performed in early stage systems engineering were removed from "systems engineering"[178]. In the B paradigm, requirements are but one of the inputs to the 'systems engineering process' (Martin, 1997) page 95; (Eisner, 1997); (Wasson, 2006) page 60; (DOD 5000.2-R, 2002), pages 83-84).

29.2.2 Process camp

Some systems engineers, particularly in INCOSE and the US DOD, are process-focused (Section 25.2). These are the campers who tend to insist that organisations must modify themselves to follow a particular process Standard. However, these campers can't seem to see the big picture and don't seem to realise that not only does the "systems engineering process" map into the ubiquitous general problem solving process but there is also currently no single widely agreed upon "systems engineering process" since over the years, the "systems engineering process" has been stated in many different and sometimes contradictory ways. These campers fail to realize that the reason why these documented processes are different is that they were developed by some entity at some point in time for a specific situation and *need to be tailored* for other specific situations. It is the systems engineer designs, architects or customizes the process that will be used to realize the solution system[179] (Chapter 19). These campers also ignore:

---

[177] The solution system is acquired to remedy a problem.

[178] This removal was documented in DOD 5000.2-R, "Mandatory Procedures for Major Defense Acquisition Programs (MDAPS) and Major Automated Information System (MAIS) Acquisition Programs," US Department of Defense, 2002.

[179] Perhaps this is because process architecting tends to be overlooked because process architecting is generally not taught in systems engineering classes which tend to

- The literature on excellence which focuses on people and ignores process (Peters and Waterman, 1982; Peters and Austin, 1985) and (Rodgers, et al., 1993).
- The axiom "garbage-in-garbage-out" (GIGO) which although originally was applied to computer data, holds true for all types of processes.
- Attempts to warn against "overemphasis on the institutionalization of processes rather than the value or effectiveness of the effort" (Armstrong, 1998).

In the last few years, the process camp has produced Model Based Systems Engineering (MBSE) by applying 21st century technology in their 20th century systems engineering process paradigm. MBSE in its current form:

- is an attempt to return to the early stage systems engineering activities performed in the 1960's and 70's , and,
- ignores the potential of the integrated information environment to produce interdependent third and fourth generation systems engineering and project management tools that could improve on the current paradigm and reduce the probability of project failures by adding expert system and artificial intelligence functionality to the tools[180]; a potential that was demonstrated at the SEEC at UniSA in the early years of the 21st century (Kasser, et al., 2002; Cook, et al., 2001; Kasser, 2000a; 2002c).

29.2.3 Problem camp

The problem solving camp can be traced back at least as far as 1980 (Gooding, 1980). These campers maintain the tradition of the pre-NCOSE systems engineers and focus on the problem and identifying the best solution available given the constraints at the time (Hitchins, 2007). Some of these campers also address carrying out that process to realize the solution system (Bahill and Gissing, 1998). This is why their systems engineering process overlaps the various versions of problem solving process (Section 25.3).

29.2.4 Discipline camp

Wymore defined systems engineering as a discipline (Wymore, 1994). Systems engineering meets the requirement for a discipline (Sections 21.2 and 21.3). However, as noted above, all the elements of the current INCOSE ap-

---

assume a process exists and start from there. Process architecting however is taught in project management classes as a part of creating project plans.

[180] As a simple example in smartening up requirements management tools, Tiger Pro contains the functionality to detect and correct some types of errors in requirements (Kasser, Tran and Matisons, 2003).

proach to systems engineering overlap those of project management and other disciplines which make it difficult to identify systems engineering as a distinct discipline for tackling complex problems. For examples, see:

- A few examples of the different overlaps between systems engineering and project management (Jenkins, 1969; Brecka, 1994; Roe, 1995; DSMC, 1996; Sheard, 1996; Johnson, 1997; Watts and Mar, 1997; Bottomly, et al., 1998; Kasser, 1996).
- Emes at al. who discussed overlaps between systems engineering and other disciplines (Emes, et al., 2005).
- Eisner who listed a general set of 28 tasks and activities that were normally performed within the overall context of large-scale systems engineering (Eisner, 1988). He calls the range of activities 'specialty skills' because some people spend their careers working in these specialties. Thus according to Eisner in 1988 [the role of][181] systems engineering overlaps at least 28 engineering specialties.
- Eisner who expanded his earlier list and discussed 30 tasks that form the central core of systems engineering (Eisner, 1997) page 156). The whole area of systems engineering management is covered in just one of the tasks. Eisner states that "*not only must a Chief Systems Engineer understand all 30 tasks; he or she must also understand the relationships between them, which is an enormously challenging undertaking that requires both a broad and deep commitment to this discipline as well as the supporting knowledge base*".
- INCOSE President John Thomas expands on this role in his presentations on the need for systems engineers with moxie, see (Thomas, 2011) for one example.
- Goode and Machol make no distinction between 'systems engineering' and 'engineering design' or even 'design' and use the terms interchangeably (Hall, 1962) page 20 citing (Goode and Machol, 1959). Archer defined design as *"a goal-directed problem solving activity"* (Archer, 1965). Fielden defined 'engineering design' as *"the use of scientific principles, technical information and imagination in the definition of a mechanical structure, machine or system to perform prespecified functions with the maximum economy and efficiency*" (Fielden, 1963) and Matchett and Briggs defined 'design' as *"the optimum solution to the sum of the true needs of a particular set of circumstances"* (Matchett and Briggs, 1966). Bahill and Dean, in discussing the requirements in the 'systems engineering process' call it the 'system design process' and use the terms 'design' and 'solution' interchangeably (Bahill and Dean, 1997). And, Hari et al. provided an example of the various activities per-

[181] Author's interpretation.

formed in new product design that overlap those of systems engineering (Hari, et al., 2004).

- The UK DERA definition of systems engineering is "*a set of activities which control the overall design, development, implementation and integration of a complex set of interacting components or systems to meet the needs of all the users*" (DERA, 1998). Controlling activities are project management activities, development and testing activities are engineering activities.
- Project management is defined as "*the planning, organizing, directing, and controlling of company resources (i.e. money, materials, time and people) for a relatively short-term objective. It is established to accomplish a set of specific goals and objectives by utilizing a fluid, systems approach to management by having functional personnel (the traditional line-staff hierarchy) assigned to a specific project (the horizontal hierarchy)*" (Kezsbom, et al., 1989). Kezsbom's systematic approach to project management requires the break down and identification of each logical subsystems component into its own assemblage of people, things, information or organization required to achieve the sub-objective (Kezsbom, et al., 1989) page 7).
- The US DOD defined Integrated Product and Process Development (IPPD) as *"a management process that integrates all activities from product concept through production/field support, using a multifunctional team, to simultaneously optimize the product and its manufacturing and sustainment processes to meet cost and performance objectives"* (DOD, 1996). Looking at industry today, Hall's mixed systems engineering teams (Hall, 1962) seem to be called IPTs and are working in the context of "concurrent engineering" which has existed as a recognizable topic since the mid 1980's. The aim of both concurrent engineering and Systems Engineering is *"to provide a good product at the right time … suitably free of defects and ready when the customer wants it*" (Gardiner, 1996)
- Configuration Management is defined as "*a field of management that focuses on establishing and maintaining consistency of a system's or product's performance and its functional and physical attributes with its requirements, design, and operational information throughout its life*" (MIL-HDBK-61A, 2001). There are two types of configuration audits within configuration management which overlap systems engineering activities (Section 28.3.2.1.3).

The discipline camp tends to account for the overlap by viewing systems engineering as a meta-discipline incorporating the other disciplines and hold that systems engineering needs to widen its span to take over other disciplines.

29.2.5 Systems thinking and non-systems thinking camps

The systems thinking camp tends to be systems engineers who can view an issue from several perspectives (Evans, 1996; McConnell, 2002; Rhodes, 2002; Martin, 2005; Selby, 2006; Beasley and Partridge, 2011), while the non-systems thinkers tend to have a single viewpoint. The non-systems thinkers also generally exhibit the 'biased jumper' level of critical thinking (Wolcott and Gray, 2003).

29.2.6 Domain systems engineering

There is a domain systems view of the role of systems engineers/engineering as well based on the role of the engineer. If you are an engineer working on a widget system then you are a widget system engineer. Examples are network systems engineers/engineering, control system engineers/engineering, communications systems engineers/engineering, hydraulic systems engineers/engineering, transportation systems engineers/systems engineering, etc.

29.2.7 Enabler camp

In the enabler camp, systems engineering is the application of holistic thinking. Moreover, it can be, and is, used in all disciplines for tackling certain types of (complex) problems; see "*[systems engineering] is a philosophy and a way of life*" (Hitchins, 1998).

29.3 Reconciling the camps

Each camp is focused on the role of the systems engineer in the workplace and each camp has a different version or vision of that role. As long as these camps were isolated, there was no problem; it is when these different roles are compared that the confusion, complexity, contradictions and overlaps with other disciplines appear. The situation is indicative of the early stages of a discipline much like the state of chemistry before the development of the periodic table of the elements or the condition of electrical engineering before the development of Ohm's Law and later the development of electrical motors before Maxwell's equations were discovered. For systems engineering to become a discipline, these camps must be reconciled or unified.

There are four ways of remedying or dealing with a problem, namely solving, resolving, dissolving or absolving the problem (Ackoff, 1978) page 13), where only the first three actually remedy the problem. The four ways are:

- **Solving the problem** is when the decision maker selects those values of the control variables which maximize the value of the outcome.

- **Resolving the problem** is when the decision maker selects values of the control variables which do not maximize the value of the outcome but produce an outcome that is good enough (satisfices the need).
- **Dissolving the problem** is when the decision maker reformulates the problem to produce an outcome in which the original problem no longer has any meaning.
- **Absolving the problem** is when the decision maker ignores the problem or imagines that it will eventually disappear on its own. Problems may be intentionally ignored because they are too expensive to remedy, or because the technical or social capability needed to provide a remedy is not known.

The chosen approach to reconciling the camps is to dissolve the problem by making a change in the paradigm. This approach redesigns the system containing the problem or changes the perspective from which the problem is viewed, to produce an innovative solution. The current systems engineering paradigm is based on the role of the systems engineer in the workplace, namely what the systems engineer does recognizing that what the systems engineer does is different in each organization. The approach to reconcile the camps is to distinguish between two systems engineering paradigms:

- **systems engineering – the role (SETR)** being what systems engineers do in the workplace, and
- **systems engineering – the activity (SETA)** that can be performed by anyone.

SETA is the set of activities known as systems engineering, while SETR is a role or job description of the systems engineer. Having made the distinction, the following criterion was used to determine if an activity does or does not belong in the set of activities known as SETA (Kasser and Hitchins, 2009; Kasser, et al., 2009):

- If the activity *deals with parts and their interactions as a whole*, then it is an activity within the set of activities to be known as SETA.
- If the activity *deals with a part in isolation*, then the activity is not an activity within the set of activities to be known as SETA but is part of another set of activities ('something else'), e.g., engineering management, software engineering, etc.

SETA is a return to Hall's definition of "*systems engineering as a function*[182] *not what a group does*[183]" (Hall, 1962) page 11). Hall added "*By recognizing the scope of the function it becomes possible to dissect it, to under-*

[182] Activity.
[183] Role.

stand its problems and to reconstruct it to make it more efficient than it is today".

Kuhn wrote that an alternative paradigm is a reconstruction of the field from new fundamentals, a reconstruction that changes some of the field's most elementary theoretical generalizations as well as many of its paradigm methods and applications (Kuhn, 1970). If SETA is an alternative systems engineering paradigm to SETR, then to meet Kuhn's requirement for an alternative paradigm it has to resolve conflicts that cannot be readily resolved within the current paradigm, namely reconcile the camps.

29.4 Testing the hypothesis

The hypothesis that SETA is a systems engineering paradigm was tested by posing the following research questions:

1. Is there another set of activities (equivalent to SETA) that can be considered as a discipline that is used in other disciplines and domains?
2. Can SETA as a discipline be differentiated from other disciplines?
3. Can the traditional SETR view of systems engineering be described in terms of SETA?

Consider each research question in turn.

29.4.1 Another set of activities.

Is there another set of activities that can be considered as a discipline that is used in other disciplines and domains?

The answer to the question is yes. Mathematics is considered both as a discipline and as a set of tools used in many if not all disciplines and domains. For example, operations research is based on mathematics, managers commonly use spreadsheets, and humanity uses the ubiquitous digital calculator to perform mathematical calculations many situations.

29.4.2 Differentiation of SETA as a discipline

Can SETA as a discipline be differentiated from other disciplines?

The answer to the question is yes which resolves the issue of differentiating systems engineering from other disciplines, something which cannot be done in the current INCOSE paradigm. SETA is often used in the form of applying systems thinking and critical thinking (Kasser and Mackley, 2008; Kasser, 2009; Kasser, 2013) in the ubiquitous generic problem-solving-solution-realization process.

Mathematics is an enabling discipline which provides a set of tools and techniques for tackling certain types of problems. Similarly SETA is not a traditional engineering discipline but can also be considered as an enabling

discipline, providing a set of tools and techniques, comprising activities that deal with parts of a system and their interactions as a whole, which are used to identify underlying problems and realize optimal solutions via the systems engineering problem solving process. This is a change in perspective with respect to the current INCOSE discipline camp which looks outwards from systems engineering. In the discipline camp, SETR is or should be taking over other disciplines. The enabler camp looks at systems engineering from the outside. From this outside perspective, SETA is an enabling discipline used in those other disciplines and professions.

Moreover, the SETA discipline is a return to the pre-NCOSE systems engineering paradigm for managing complexity and innovation as documented by the literature of the time including (Goode and Machol, 1959) and (Hall, 1962), when 'systems engineering' (SETA) was a tool used by, or synonymous with, 'design' (Goode and Machol, 1959; Hall, 1962; Fielden, 1963; Archer, 1965; Matchett and Briggs, 1966; Jones, 1970) page 115) and 'systems engineers' (SETR) performed SETA using systems engineering tools (Wilson, 1965; Alexander and Bailey, 1962; Chestnut, 1965) such as:

- Probability,
- Single thread – system logic,
- Queuing theory,
- Game theory,
- Linear programming,
- Group dynamics,
- Simulation, and
- Information theory

29.4.3 Traditional activities

Can the traditional SETR view of systems engineering be described in terms of SETA?

The answer is yes. From the Big Picture perspective, SETA and non-SETA are subsets of the whole set of workplace activities performed in the problem-solving-solution-realization process which covers the entire activity from the time an issue is raised (which in turn becomes a problem which then needs a solution) to the time of disposal of the solution system when it no longer satisfies the need. This set of activities may be partitioned into different mixes of subsets in various ways such as by profession and discipline (project/engineering management, systems engineering, engineering, new product design, etc.) and by time (the phases in the system lifecycle). These SETA and non-SETA activities can be also be grouped into the three interdependent streams of work, Development, Management and Test/Quality which merge at predefined milestones during the SDLC as shown Figure 2-2.

Due to the various ways in which SETA and non-SETA have been allocated to personnel[184] performing SETR and non-SETR, in any specific organisation at any specific time, a specific SETR will perform a mixture of SETA and non-SETA in one or more of the three interdependent steams of work. For example, one instance of SETR might perform systems architecting in the development stream while another may perform systems engineering management in the management stream and a third perform system integration. At the same time non-SETR personnel, such as designers or project managers might be performing different mixtures of non-SETA and SETA. This situation also explains why there seem to be a lot of people in industry doing SETA without being aware of the term 'systems engineering'.

SETR has traditionally been associated with Defence and aerospace due to their conflation. MIL-STD-499 defines a total systems approach for the development of defence systems. Section 1.1 of the standard states that "*the systems engineering process is applied iteratively through the system lifecycle to translate stated problems into design requirements*" (MIL-STD-499B, 1993). Yet the systems engineering process described in the standard is just a version of the latter part of ubiquitous generic problem-solving-solution-realization process stated in engineering language. SETA can thus be considered as an enabling discipline used in the problem-solving-solution-realization process performed in the domain of acquiring and developing Defence systems. ISO/IEC 15288 also contains a list of processes used in the domain of acquiring and developing systems which overlap all three streams of work (Arnold, 2002). Each of those processes contains SETA and non-SETA.

29.5 Discussion

Systems engineering has been associated with Defence and aerospace due to their conflation. SETA is often used to perform design, yet Love argues "*the central activity of designing is 'epistemologically different' from the application of systems methods, techniques, and approaches and perspectives".* It [his paper] *"suggests the uncritical conflation of the activities of designing and systems analysis seriously compromises theoretical and practical developments in both Systems and Design and this has led to confusion in both fields and to the development of extensive, unnecessary and over-complex theories targeting an epistemologically irresolvable problem*" (Love, 2003). Overly complex theories are a symptom of a flaw in the paradigm. There is a need to differentiate SETA from the system acquisition and development

[184] The word 'personnel' is used to avoid the semantically loaded terms engineers, systems engineers, project manager, etc.

lifecycle currently conflated with systems engineering.

Systems engineering began as SETA and evolved into SETR (Chapter 23). The SETR paradigm, led by the discipline camps looks outwards from systems engineering in an effort to expand SETR into a meta-discipline. The SETA paradigm on the other hand, looks inwards at systems engineering from the outside and sees SETA as an enabling discipline applied inside those other disciplines. SETR is performed in all columns of the HKMF shown in Figure 21-3. However, in any one layer, most of SETA tends to be performed in Column's A and B; the activities involved in figuring out the problem and determining and specifying the optimal solution system to be realized. Most of the activities in columns C, D and E are non-SETA performed by engineers, designers and testers. Some SETA does occur to ensure that the system level specifications are met and to deal with the emergent properties. SETA picks up again during systems integration and the commissioning of the solution system in the field phase of the SDLC in column F. In addition, if the subsystems are complicated enough they may have their own set of SETA and non-SETA. For example, in the Apollo program, there was an overall set of SETA for the entire mission; yet the realization of each of the subsystems was non-SETA engineering as far as the entire mission was concerned. However, the realization of the Lunar Module and the ALSEP and the other tier one subsystems each needed SETA and non-SETA activities. And within the ALSEP, realization of the central station and each of the scientific experiments required SETA and non-SETA activities irrespective of the job title (SETR) of the person who performed the activities.

SETA does not actually produce a tangible product. SETA produces documents, namely the plans, specifications, reports, etc. during the SDLC. The non-SETA activities including engineering actually produce the solution system.

The SETA/SETR paradigms provide for agreement on SETA[185] and the recognition of the reasons for the different roles of the systems engineer. The lifecycle camp views SETR over the entire problem-solving-solution-realization process. The process camp views SETR in the latter part of the SLC documented in the various standards applied to systems engineering (Honour and Valerdi, 2006; Haskins, 2011), and the problem camp views SETR as a problem solving role anywhere in the problem-solving-solution-realization process.

The SETA/SETR paradigms also provide a solution to the problem of developing a manageable SEBoK. SETR has evolved over time so that it is different in practically every organisation and has various degrees of overlap

[185] The INCOSE Fellows accepted the definition in 2009, see (Kasser and Hitchins, 2009).

with the roles of other disciplines. This makes differentiating systems engineering from the other roles in the workplace extremely difficult and has resulted in the discipline camp calling systems engineering a meta-discipline that embodies the others. It also complicates developing a manageable SEBoK, since in order to be complete the contents of the SEBoK-SETR would have to cover the knowledge needed in the different SETR in practically every organisation and the knowledge needed in the disciplines where the overlaps occur. SETA on the other hand, can readily be determined in an objective manner by examining the activities in all three streams of work in the SDLC in each column of the HKMF, sorting out the SETA and creating a SETA-SEBoK with traceability to each area in the HKMF. Moreover, additional research can tag each of the areas of the HKMF with required competencies.

Personnel known as systems engineers (SETR) often perform a mixture of SETA and engineering. SETA in all three streams of work incorporates the mental activities of applying holistic thinking (Kasser, 2013). This is the way post-independence Singapore was systems engineered, by personnel in public health, housing, Defence, transportation, etc. and is the essence of the SETA paradigm irrespective of SETR and domain. The people who do SETA do it as a way of life (Hitchins, 1998) whether they are, or are not, known as systems engineers (SETR). For example, SETA is used when:

- **Cooking a meal.** The meal emerges from both the process and the combination of, and the interaction between, the ingredients. The best ingredients will not save a meal that was over-cooked or under-cooked.
- **Diagnosing an illness.** Good physicians consider the symptoms holistically in the context of the physiology of the patients and their environments.
- **Organising a conference.** The conference emerges from the combination of, and interaction between, the location, speakers, reviewers, delegates, and other entities.
- **Solving a crime.** Detectives, upon investigation, find a variety of clues which (should) lead to the perpetrator

And the personnel who perform these activities are not known as systems engineers. This is not surprising since the need for systems thinking in tackling problems has also long been recognised outside the systems engineering community: for example "*when people know a number of things, and one of them understands how the things are systematically categorized and related, that person has an advantage over the others who don't have the same understanding"* (Luzatto, circa 1735). Ford discussed looking at the value chain of products transported by the railroads as a system, in order to solve the transportation problem of his time (Ford and Crowther, 1922), pages 230-231). Other examples are Crosby's "completeness" (Crosby, 1979), Deming's "system of profound knowledge" (Deming, 1993), and Sen-

ge's "fifth discipline" (Senge, 1990); all state the need for systems thinking, and the benefits to be gained therefrom[186].

29.6 Conclusions

Systems engineering can be considered as being two paradigms:

1. **SETR:** systems engineering performed by personnel known as systems engineers. Examples are network systems engineering, control system engineering, communications systems engineering, etc. In many instances the type of system is dropped from the title. This systems engineering overlaps other disciplines and the exact role depends on the situation.
2. **SETA:** the problem identification and solution realization activities on a system at the system level in accordance with the activity definition (Kasser and Hitchins, 2009). This systems engineering is an enabling discipline like mathematics.

Separation of the SETA and SETR paradigms:

- Resolves the conflicts and contradictions in the current state of systems engineering; in addition:
  1. The traditional activities known as systems engineering can be described in terms of SETA and SETR.
  2. SETR is the job title for a person who performs a mixture of SETA and non-SETA.
  3. SETA is an enabling discipline that is used in other disciplines and domains.
  4. SETA as a discipline can be differentiated from other disciplines.
- Resolves issues due to the overlap between systems engineering and project management.

Irrespective of what they are called[187], personnel leading projects should be Type Vs (Section 23.3) with sufficient knowledge and experience in the problem, solution and implementation domains to be able to make informed decisions and understand the advice of experts in each domain. They should also be people who apply SETA all the time, namely it is their way of life.

---

[186] Systems thinking lets you understand the situation, however you need to go beyond systems thinking and use holistic thinking to create innovative solutioms to complex problems (Kasser, 2013).

[187] The term engineer-leader is used in Singapore to indicate that the person needs proficiency in both systems engineering and project management.

Figure 29-1 The Type V systems engineer

2013

30 Postscript

This book has viewed systems engineering from several perspectives applying holistic thinking to the problem of determining the nature of systems engineering. The output product of the research has been

- The HKMF for understanding systems engineering. The benefits of the HKMF problem-based paradigm have been discussed in the relevant Chapters. Other benefits not previously discussed should include:
    - Moving organizations away from activities (industrial age) and towards problem solving and the knowledge needed to make informed decisions (information age).
    - Requiring the education of a generation of people who will understand the consequences of their decisions. This could be the most tangible benefit and the real change.
- The holistic thinking perspectives (HTP) introduced in section 28.2 are expanded in Holistic Thinking: creating innovative solutions to complex problems (Kasser, 2013).
- An understanding of why systems engineering seems to overlap other disciplines and is applicable in all domains. **Systems engineering is an enabling discipline providing a set of thinking tools for creating solutions to complex problems**.

Perceived from the Temporal perspective, the body of knowledge that we have today is not the sum of all knowledge. For example, McCloskey wrote "*In the present century, physical scientists, aided by powerful mathematical tools, have reduced to principle many phenomena little known and less well understood by the predecessors*" (McCloskey, 1954) page 258. One could argue that these scientists have solved what appeared at the time to be wicked problems? What are we going to learn in the future and how will it change our way of thinking?

31 Acronyms

ADATS	Air-Defense Anti-Tank System
ALSEP	Apollo Lunar Surface Experiments Package
ASQ	American Society for Quality
BPR	Business Process Reengineering
CAIV	Cost as an independent variable
CASE	Computer Aided Software Engineering
CCB	Configuration Control Board
CCS	Central control station (LuZ)
CES	Control and electronics system (Luz)
CEST	Capacity for engineering system thinking
CMM	Capability Maturity Model
CMMF	Competency model maturity framework
CONOPS	Concept of operations
COTR	Contracting Officer's Technical Representative
COTS	Commercial off the shelf
CSEP	Certified systems engineering professional
CRIP	Categorized Requirements in Process
DERA	[UK] Defence Evaluation and Research Agency
DIVAD	Division Air Defense
DOD	Department of Defense
DODAF	Department of Defense Architecture Framework
DT&E	Developmental test and evaluation
DR	Discrepancy report
FRAT	Functions Requirements Answers and Test
FRE	Framework for Research and Education
FREDIE	Framework for Requirements Engineering in a Digital Integrated Environment
HTP	Holistic thinking perspective
HKMF	Hitchins-Kasser-Massie Framework for understanding systems engineering
ICPM	Institute of Certified Professional Managers
IDE	Integrated Digital Environment
IEEE	Institute of Electrical and Electronics Engineers
INCOSE	International Council on Systems Engineering
IPPT	Integrated Product Process Team
IPT	Integrated Process Teams
ISO	International Organization of Standards
IT	Information Technology
IV&V	Independent Verification and Validation
JSF	Joint Strike Fighter

LEO	Low earth orbiting
LOC	Local controller (LuZ)
LSA	Logistics Support Analysis
MATO	Multiple-award-task-ordered
MBNQA	Malcolm Baldridge National Quality Award
MBWA	Management by Walking Around"
MTBF	Mean Time Between Failures
MTTR	Mean Time to Repair
NASA	The National Aeronautics and Space Administration
NCOSE	National Council on Systems Engineering
NDIA	National Defense Industrial Association
NST	Network Scheduling Tool
O&M	Operations and maintenance
OCD	Operations Concept Document
ODC	Other Direct Charges
OOCONOPS	Object-oriented CONOPS
OOM	Object-oriented Methodology
OT&E	Operational test and evaluation
PAM	Product Activity Milestone
PDCA	Plan Do Check Act
PDR	Preliminary Design Review
PERCY	PERsonal portal into Cyberspace
PERT	Program Evaluation and Review Technique
PIT	Process improvement team
PPPT	People Process Product Time
PSP	Personal Software Process
QSE	Quality System Elements
RFP	Request for Proposal
ROI	Return on investment
RRS	Reward and recognition system
RTM	Requirements traceability matrix
SBA	Small Business Administration
SCADC	Standard Central Air Data Computer
SCCB	Strategic CCB
SDB	Small Disadvantaged Business
SDLC	Software and Systems Development Lifecycle
SDR	System Design Review
SEBoK	Systems engineering body of knowledge
SECF	Systems Engineering Competency Framework
SECT	Systems engineering competency taxonomy
SEEC	Systems Engineering and Evaluation Centre
SEGS	Solar electrical power generating system
SEMP	Systems Engineering Management Plan
SETA	Systems engineering – the activity

SETR	Systems engineering – the role
SLC	System and software lifecycle
SOW	Statement of Work
SSM	Soft system methodology
SRD	System Requirements Document
SRR	System Requirements Review
STOVL	Short Take-Off & Vertical Landing
T&E	Test and Evaluation
TEMP	Test and Evaluation Master Plan
TIGER	Tool to ingest and elucidate requirements
TQM	Total Quality Management
TSI	Total Systems Intervention
UML	Unified Modelling Language
UMUC	University of Maryland University College
UniSA	University of South Australia
US	United States [of America]
USAF	US Air Force
USMC	US Marine Corps
USN	US Navy
WBS	Work Breakdown Structure

32 References

The papers cited as references to Kasser's work may generally be found on the publications page at http://therightrequirement.com.

Ackoff, R. L., *The Art of Problem Solving*, John Wiley & Sons, New York, 1978.

Ackoff, R. L., "The Future of Operational Research is Past," *Critical Systems Thinking Directed Readings*, R. L. Flood and M. C. Jackson (Editors), 1991.

---, *Ackoff's Best. His Classic Writings on Management*, John Wiley & Sons, Inc., New York, 1999.

Adams, S., Dilbert cartoons, 2006, http://www.dilbert.com, accessed on 6 November 2006.

Alderfer, C. P., *Existence, Relatedness and Growth: Human Needs in Organizational Settings*, The Free Press, 1972.

Alexander, I. F. and Stevens, S., *Writing Better Requirements*, Addison-Wesley, 2002.

Alexander, J. E. and Bailey, J. M., *Systems Engineering Mathematics*, Prentice-Hall, Inc., Englewood Cliff, NJ., 1962.

Allen, M., *Smart thinking: skills for critical understanding and writing*, Oxford University Press, 2004.

Allison, J. S. and Cook, S. C., 1998, *The New Era in Military Systems Thinking and Practice*, proceedings of First Regional Symposium of the Systems Engineering Society of Australia INCOSE Region 6 (SETE 98).

American Heritage, *The American Heritage Dictionary of the English Language*, Houghton Mifflin Company, 2000.

Anderson, K. and Dibb, P., "Strategic Guidelines for enabling research and development to support Australian Defence, paragraph 121," 1996.

Anderson, L. W., Krathwohl, D. R., Airasian, P. W., Cruikshank, K. A., Mayer, R. E., Pintrich, P. R., Raths, J. and Wittrock, M. C. (Editors), *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*, Allyn and Bacon, 2000.

Angel, C. D. and Froelich, J., Six Sigma: What Went Wrong?, 2008, http://www.destinationcrm.com/Articles/Columns-Departments/The-Tipping-Point/Six-Sigma-What-Went-Wrong-51394.aspx, accessed on 26 April 2009.

ANSI/EIA-632, "Processes for Engineering a System," American National Standards Institute and Electronics Industries Association, Arlington, VA, 1999.

Archer, L. B., *Systematic Method for Designers*, Council of Industrial Design, London, 1965.

Armstrong, J. R., 1998, *How Maturity Modeling Saved My Softball Team*, proceedings of the 8th INCOSE International Symposium.

Arnold, S., 2000, *Systems Engineering: From Process towards Profession*, proceedings of The 10th Annual Symposium of the INCOSE.

---, "ISO 15288 Systems engineering — System lifecycle processes," International Standards Organisation, 2002.

Arthur, L. J., *Rapid Evolutionary Development*, John Wiley & Sons, Inc., 1992.

---, *Improving Software Quality An Insider's Guide to TQM*, John Wiley & Sons, Inc., 1993.

Aslaksen, E. W., *System Thermodynamics: A Model Illustrating Complexity Emerging from Simplicity*, Systems Engineering 7 (2004), no. 3, 271-284.

Aslaksen, E. W. and Belcher, R., *Systems Engineering*, Prentice Hall, New York, 1992.

Au, T. and Stelson, T. E., *Introduction to Systems Engineering Deterministic Models*, Addison-Wesley Publishing Company, 1969.

Avison, D. and Fitzgerald, G., *Information Systems Development: Methodologies, Techniques and Tools*, McGraw-Hill Education (UK), Maidenhead, 2003.

Badaway, M., *Educating Technologists in Management of Technology*, EMR (1995), no. Fall.

Bahill, A. T. and Dean, F. F., 1997, *The Requirements Discovery Process*, proceedings of the 7th International Symposium of the INCOSE.

Bahill, A. T. and Gissing, B., *Re-evaluating systems engineering concepts using systems thinking*, IEEE Transaction on Systems, Man and Cybernetics, Part C: Applications and Reviews 28 (1998), no. 4, 516-527.

Bar-Yam, Y., 2003, *When Systems Engineering Fails --- Toward Complex Systems Engineering*, proceedings of Systems, Man and Cybernetics, 2003. IEEE International Conference on.

Barry, K., Domb, E. and Slocum, M. S., TRIZ - What Is TRIZ?, 2007, http://www.triz-journal.com/archives/what_is_triz/, accessed on 31 October 2007.

Bass, L., Clements, P. and Kazman, R., *Software Architecture in Practice*, Addison Wesley, 1998.

Beasley, R. and Partridge, R., 2011, *The Three Ts of Systems Engineering - Trading, Tailoring and Thinking*, proceedings of the 21st Annual Symposium of the INCOSE.

Beer, S., *Brain of the Firm - A development in Management Cybernetics*, Herder and Herder, NY, 1972.

---, *The Heart of Enterprise*, John Wiley & Sons, Stafford Beer Classic Edition, Chichester, 1994.

Bender, W. G., *Systems Engineering Reading List, 62-1*, Bell Telephone Laboratories, Nutley, NJ, 1962.

Bentley, C., *PRINCE 2 A Practical Handbook*, Oxford: Butterworth Heinemann, 1997.

Berg, J. M., Levia, O. and Rouillard, J., *Object-Oriented Modelling*, Kluwer Academic Publishers, 1996.

Biemer, S. M. and Sage, A. P., "Systems Engineering: Basic Concepts and Life Cycle," *Agent-Directed Simulation and Systems Engineering*, L. Yilmaz and T. Oren (Editors), Wiley-VCH, Weinheim, 2009.

Biggs, J., *Teaching for Quality Learning in University*, Society for Research into Higher Education and Open University Press, 1999.

Blanchard, B., *System Engineering Management*, John Wiley & Sons, Inc., 1998.

Blanchard, B. B. and Fabrycky, W., *Systems Engineering and Analysis*, Prentice Hall, 1981.

---, *Systems Engineering and Analysis*, Pearson Prentice Hall, Upper Saddle River, NJ, 2006.

Boehm, B., *A Spiral Model of Software Development and Enhancement*, IEEE Computer (1988), no. May.

---, *Integrating Software Engineering and Systems Engineering*, Systems Engineering 1 (1994), no. 1.

Boehm, B., *Unifying SWE and SE*, Computer (2000), no. March.

Bottomly, P. C., Brook, P., Morris, P. W. and Stevens, R., 1998, *A Study of the Relationship of Systems Engineering to Project Management*, proceedings of Fourth Annual Symposium of the INCOSE-UK.

Brecka, J., *Sabotage! Survey Finds Internal Resistance to Quality Initiatives*, Quality Progress (1994).

Brekka, L. T., Picardal, C. and Vlay, G. J., 1994, *Integrated Application of Risk Management and Cost of Quality*, proceedings of The 4th Annual International Symposium of the NCOSE.

Brill, J. H., *Systems Engineering ---A Retrospective View*, Systems Engineering 1 (1998), no. 4, 258-266.

Britton, C. and Doake, J., *Software System Development*, McGraw-Hill Education (UK) Ltd., 2003.

Brodie, E. J., *Software Engineering An Object-Oriented Perspective*, John Wiley & Sons, Inc., 2001.

Brooks, B. and Mawby, D., 1998, *Better Decision Making for Complex Engineering Projects*, proceedings of Fourth Annual Symposium of the INCOSE-UK.

Brooks, F., *The Mythical Man-Month Essays on Software Engineering, Reprinted with corrections*, Addison-Wesley Publishing Company, 1982.

Bruno, M. E. and Mar, B. W., 1997, *Management of the Systems Engineering Discipline*, proceedings of the 7th Annual Symposium of the International Council on Systems Engineering.

Budd, T., *An Introduction to Object-Oriented Programming*, Addison-Wesley, 1996.

Buede, D. M., *The Engineering Design of Systems*, John Wiley & Sons, Inc., 2000.

Bungay, S., *The Most Dangerous Enemy*, Aurum Press, London, England, 2000.

Buren, J. V. and Cook, D. A., *Experiences in the Adoption of Requirements Engineering Technologies*, Crosstalk - The Journal of Defense Software Engineering (1998), no. December.

Burke, G. D., Harrison, M. J., Fenton, R. E. and Carlock, P. G., 2000, *An approach to develop a systems engineering curriculum for human capital and process improvement*, proceedings of the10th Annual International Symposium of the INCOSE.

Caltrans, System Engineering Guidebook for Intelligent Transportation Systems Version 2.0, California Department of Transportation, Division of Research and Innovation, 2007.

Campbell, J. W., "Editorial," *Analog*, 1960.

CareerOneStop, Using Competency Models, 2011, http://www.careeronestop.org/competencymodel/learncm.aspx, accessed on 11 November 2011.

Carroll, L., *Through the Looking Glass*, 1872.

Carson, R. S., 2001, *Keeping the Focus During Requirements Analysis*, proceedings of the 11th International Symposium of the International Council on Systems Engineering.

Chamberlain, R. G. and Shishko, R., 1991, *Fundamentals of Systems Engineering at NASA*, proceedings of INCOSE/ASEM Proceedings.

Chang, C. K., Cleland-Huang, J., Hua, S. and Kuntzmann-Combelles, A., *"Function-Class Decomposition", A Hybrid Software Engineering Method*, IEEE Computer (2001), 87-93.

Chang, C. K. and Hua, S., 1994, *A New Approach to Module-Oriented Design of Object Oriented Software*, proceedings of 18th International Computer Software and Applications Conference, (COMPSAC94).

CHAOS, The Chaos study, The Standish Group, 1995, http://www.standishgroup.com/chaos.html, accessed on March 19, 1998.

---, Chaos Chronicles, The Standish Group, 2004.

Chapanis, A., "Human Engineering," *Operations Research and Systems Engineering*, C. D. Flagle, W. H. Huggins and R. H. Roy (Editors), Johns Hopkins Press, Baltimore, 1960.

Chapman, W. L., Bahill, A. T. and Wymore, A. W., *Engineering Modeling and Design*, CRC Press, Boca Raton, Fla., 1992.

Chase, R. B., Aquilano, N. J. and Jacobs, F. R., *Productions and Operations Management*, Irwin McGraw-Hill, 1998.

Chase, R. B. and Stewart, D. M., *Make Your Service Fail-Safe*, Sloan Management Review 35 (1994), no. 3, 35-44.

Checkland, P., *Systems Thinking, Systems Practice*, vol. Chichester, John Wiley & Sons, 1991.

Checkland, P. and Holwell, S., *Information, Systems and Information Systems: making sense of the field*, vol. Chichester, John Wiley & Sons, 1998.

Checkland, P. and Scholes, J., *Soft Systems Methodology in Action*, John Wiley & Sons, 1990.

Chestnut, H., *Systems Engineering Tools*, John Wiley & Sons, Inc., New York, 1965.

Churchman, C. W., *The Systems Approach and its Enemies*, Basic Books, Inc., New York, 1979.

Churchman, C. W., Ackoff, R. L. and Arnoff, E. L., *Introduction to Operations Research*, Wiley, 1957.

Clausing, D., *Total Quality Development*, ASME Press, 1994.

Clausing, D. and Cohen, L., 1994, *Recent Developments in QFD in the United States*, proceedings of Institution of Mechanical Engineering Conference.

CMM, *Carnegie Mellon University, The Capability Maturity Model: Guidelines for Improving The Software Process*, Addison-Wesley, 1995.

Cook, S. C., 2000, *What the Lessons Learned from Large, Complex, Technical Projects Tell us about the Art of Systems Engineering*, proceedings of the International Symposium of the INCOSE.

---, 2001, *On the Acquisition of Systems of Systems*, proceedings of the 11th annual Symposium of the INCOSE.

---, *Principles of Systems Engineering - Course Notes*, Systems Engineering and Evaluation Centre, University of South Australia, Adelaide, 2003.

Cook, S. C., Kasser, J. E. and Asenstorfer, J., 2001, *A Frame-Based Approach to Requirements Engineering*, proceedings of 11th International Symposium of the INCOSE.

Cook, S. C., Kasser, J. E. and Ferris, T. L. J., 2003, *Elements of a Framework for the Engineering of Complex Systems*, proceedings of the 9th ANZSYS Conference.

Costello, R. B., Bolstering Defense Industrial Competitiveness: Preserving Our Heritage, the Industrial Base Securing Our Future, Office of the Undersecretary of Defence (Acquisition), 1988.

Covey, S. R., *The Seven Habits of Highly Effective People*, Simon & Schuster, 1989.

Cox, C., Welcoming Immigrants to a Diverse America: English as our Common Language of Mutual Understanding, 1996, http://policy.house.gov/documents/statements/english.htm, accessed on 10 August 2000.

Crosby, P. B., *Quality is Free*, McGraw-Hill, 1979.

Crosby, P. B., *The Art of Getting Your Own Sweet Way*, McGraw-Hill Book Company, 1981.

---, *Completeness: Quality for the 21st Century*, Dutton, New York, 1992.

D'Souza, D. F. and Willis, A. C., *Objects, Components, and Frameworks With UML: The Catalysis Approach*, Addison-Wesley, 1998.

Daniels, J., Bahill, T. and Botta, R., "A Hybrid Requirements Capture Process," *the 15th annual International Symposium of the INCOSE*, Rochester, NY., 2005.

Darke, P. and Shanks, G. G., *User Viewpoint Modelling: understanding and representing user viewpoints during requirements definition*, Information Systems Journal 7 (1997), no. 3, 213-219.

DAU, Defense Acquisition University (DAU). SPRDE-SE/PSE competency model, 4/14/10 version, 2010, https://acc.dau.mil/CommunityBrowser.aspx?id=315691&lang=en-US, accessed on

Davidow, W. H. and Malone, M. S., *The Virtual Corporation*, Edward Burlingame Books/Harper Business, 1992.

Davies, J., 1998, *Making Choices at the Right Time: Producing Better Systems*, proceedings of Fourth Annual Symposium of the INCOSE-UK.

Davis, S., 2003, *FCS System of Systems' Engineering and Integration*, proceedings of NDIA Systems Engineering Conference.

Deevy, E., *Creating the Resilient Organization. A RapidResponse Management Program*, Prentice Hall Inc., New Jersey, 1995.

DeMarco, T., *Why Does Software Cost So Much? And Other Puzzles of the Information Age*, Dorset House Publishing, 1995.

DeMarco, T. and Lister, T., *Peopleware Productive Projects and Teams*, Dorset House Publishing, 1999.

Deming, W. E., *Out of the Crisis*, MIT Center for Advanced Engineering Study, 1986.

Deming, W. E., *The New Economics for Industry, Government, Education*, MIT Center for Advanced Engineering Study, 1993.

Dennison, K., 2000, *Integration or Partnership - Trends in UK MoD Air System Test and Evaluation*, proceedings of Fourth Test and Evaluation International Aerospace Forum.

Denzler, D. and Kasser, J. E., 1995, *Designing Budget Tolerant Systems*, proceedings of the 5th Annual International Symposium of the NCOSE.

DERA, DERA Systems Engineering Practices Reference Model, DERA/LS(SEC-FH)/PROJ/018/G01, Defence Evaluation and Research Agency (DERA), 1997.

---, *Systems Engineering has a Promising Future*, DERA News (1998), no. January.

Dewitz, S. D., *Systems Analysis and Design and the Transition to Objects*, Irwin/McGraw Hill, 1996.

DOD 5000.2-R, Mandatory Procedures for Major Defense Acquisition Programs (MDAPS) and Major Automated Information System (MAIS) Acquisition Programs, US Department of Defense, 2002.

DOD, DoD Guide to Integrated Product and Process
Development (Version 1.0),, DOD, 1996.

---, National Security Space Strategy Unclassified Summary, United States Department of Defense, 2011.

DOD IPPD, *DoD Integrated Product and Process Development Handbook*, Office of the Undersecretary of Defense (Acquisition and Technology), Washington, DC., 1998.

DoDAF, *DoD Architecture Framework Version 1.0, 9 February 2004*, 2004.

Dolan, T., *Best Practices In Process Improvement*, Quality Progress (2003), no. August 2003, 23-28.

Donaldson, S. E. and Siegel, S. G., *Cultivating Successful Software Development A Practitioner's View*, Prentice Hall PTR, Upper Saddle River, New Jersey, 1997.

Dorfman, M. and Thayer, R. H., *System and Software Requirements Engineering*, IEEE Computer Society Press, 1990.

Dreyfus, H. and Dreyfus, S., *Mind over machine: The power of human intuition and expertise in the era of the computer*, Free Press, New York, 1986.

Drucker, P. F., *Management: Tasks, Responsibilities, Practices*, Harper & Roe, New York, 1973.

---, *Managing in Turbulent Times*, Harper Row Publishers, 1980.

---, *Innovation and Entrepreneurship*, Harpercollins, 1985.

Drucker, P. F., *Managing for the Future. The 1990s and Beyond*, Truman Talley Books/Plume, New York, 1993.

Drucker, P. F., *Managing in a time of Great Change*, Truman Talley Books/Dutton, New York, 1995.

DSMC, *Systems Engineering Management Guide*, vol. May 1996, Defence Systems Management College, 1996.

Eckman, D. P. (Editor), *Systems: Research and Design*, John Wiley & Sons, Inc., 1961.

EIA 632, "EIA 632 Standard: Processes for engineering a system," 1994.

Eichhorn, R., Developing thinking skills: critical thinking at the army management staff college, 2002, http://www.amsc.belvoir.army.mil/roy.html, accessed on April 11, 2008.

Eisner, H., *Computer Aided Systems Engineering*, Prentice Hall, 1988.

---, *Essentials of Project and Systems Engineering Management*, John Wiley & Sons, Inc., New York, 1997.

El_Emam, K. and Hoeltje, D., *Qualitative Analysis of a Requirements Change Process*, Empirical Software Engineering 2 (1997), 143-207.

El_Emam, K. and Madhavji, N., *An Instrument for Measuring the Success of the Requirements Engineering Process in Information Systems Development*, Empirical Software Engineering 1 (1996), 201-240.

Emes, M., Smith, A. and Cowper, D., *Confronting an identity crisis - How to brand systems engineering*, Systems Engineering 8 (2005), no. 2, 164-186.

Endres, A. and Rombach, D., *A Handbook of Software and Systems Engineering*, Pearson Education Ltd., 2003.

Ennis, M. R., Competency Models: A Review of the Literature and The Role of the Employment and Training Administration (ETA), U. S. Department of Labor, 2008, http://www.careeronestop.org/competencymodel/info_documents/OPDRLiteratureReview.pdf, accessed on 8 September 2011.

ETA, General Competency Model Framework, The Employment and Training Administration, 2010, http://www.careeronestop.org/competencymodel/pyramid.aspx, accessed on 9 September 2011.

Evans, D. C., Systems Engineering Lessons Learned (SELL), 1989, http://evansopticalengineering.com/Page00/sysengll.htm, accessed on 6 September 2011.

Evans, R. P., 1996, *Engineering of Computer-based Systems (ECBS) Three New Methodologies--Three New Paradigms*, proceedings of the 6th International Symposium of the INCOSE.

Fabrycky, W., "International Systems Engineering Programs," *the INCOSE Education and Research Working Group Meeting, 29 June 2003*, Washington, USA, 2003.

Fanmuy, G., 2004, *Best practices for drawing up a requirements baseline - P192*, proceedings of the 14th International Symposium of the INCOSE.

Farnham, D. T., *Executive Statistical Control*, vol. 6, Industrial Extension Institute, New York, 1920.

Faulconbridge, R. I. and Ryan, M. J., 1999, *A Framework for Systems Engineering Education*, proceedings of Systems Engineering Test and Evaluation (SETE) Conference.

---, *Managing Complex Technical Projects: A Systems Engineering Approach*, Artech House, Boston, 2003.

Fayol, H., *General and Industrial Management*, Sir Isaac Pitman and Sons, Ltd., London, 1949.

Feigenbaum, D. S., *Systems Engineering - A Major New Technology*, Industrial Quality Control 20 (1963), no. September 1963, 9-13.

Fielden, G. B. R., ('The Fielden Report') Engineering Design, Her Majesty's Stationary Office, 1963.

Fisher, J., 1996, *Defining the Roles and Responsibilities of the Systems Engineering Organization/Team*, proceedings of the 6th Annual Symposium of the International Council on Systems Engineering.

Flood, R. L. and Jackson, M. C., *Creative Problem Solving*, Wiley, 1991.

Florida, Writing a Project Systems Engineering Management Plan, Florida Department of Transportation Traffic Engineering and Operations Office, 2006, http://www.floridaits.com/SEMP/Files/PDF_Report/060929-PSEMP-V4.pdf, accessed on 14 June 2010.

FM_770-78, Field Manual: System Engineering, Headquarters, Department of the Army, 1979.

Ford, H. and Crowther, S., *My Life and Work*, Doubleday Page & Company, New York, 1922.

Forsberg, K. and Mooz, H., 1991, *The Relationship of System Engineering to the Project Cycle*, proceedings of Annual Conference of the National Council on Systems Engineering, National Council on Systems Engineering.

Forsberg, K., Mooz, H. and Cotterman, H., *Visualising Project Management*, John Wiley & Sons, Inc., 2000.

Fowler, M., *Analysis Patterns: Reusable Object Models*, Addison-Wesley, 1997.

Frank, M., *Knowledge, Abilities, Cognitive Characteristics and Behavioral Competences of Engineers with High Capacity for Engineering Systems Thinking (CEST)*, The INCOSE Journal of Systems Engineering 9 (2006), no. 2, 91-103.

Frank, M., *Assessing the interest for systems engineering positions and the Capacity for Engineering Systems Thinking (CEST)*, Systems Engineering 13 (2010), no. 2.

Frank, M. and Waks, S., *Engineering systems thinking: A multifunctional definition*, Systemic Practice and Action Research 14 (2001), no. 3, 361-379.

Friedman, G. J., *On the Unification of Systems Engineering*, INCOSE INSIGHT 8 (2006), no. 2, 16-17.

Frosch, R. A., *A new look at systems engineering*, IEEE Spectrum (1969), no. September, 25.

Frosh, R. A., *A new look at systems engineering*, IEEE Spectrum (1969), no. September.

FS-1037C, Federal Standard 1037C, 1996, http://www.its.bldrdoc.gov/fs-1037/fs-1037c.htm, accessed on July 1, 2004.

Gabb, A., 2001, *Front-end Operational Concepts - Starting from the Top*, proceedings of The 11th Annual Symposium of the INCOSE.

Gabb, A., Caple, G., Haines, D., Lamont, D., Davies, P., Hall, A., Van Gaasbeek, J., Eppig, S., Jones, D. and Vietinghoff, B., 2001, *Requirements Categorization*, proceedings of the 11th Annual Symposium of the INCOSE.

Gabbar, H. A., Shimada, Y. and Sukuzi, K., *Object Interface System Using a Plan Object-Oriented Model*, Systems Engineering 4 (2001), no. 3.

Gallagher, B., Wilson, O. R. and Levinson, J. C., *Guerrilla Selling*, Houghton Mifflin Company, 1992.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J., *Design Patterns, Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

GAO, DEFENSE ACQUISITIONS Major Weapon Systems Continue to Experience Cost and Schedule Problems under DOD's Revised Policy, GAO, 2006.

Gardiner, G., *Concurrent and Systems Engineering same thing, different name, or are they both just new product introduction?*, IEE Engineering Management Journal (1996), no. February.

Gavito, V., Verma, D., Dominick, P., Pennotti, M., Giffin, R., Barrese, T. and et al., Technical leadership development program, Systems Engineering Research Center (SERC), Stevens Institute of Technology, Hoboken, NJ., 2010.

GDRC, The Problem Solving Process, 2009, http://www.gdrc.org/decision/problem-solve.html, accessed on 11 Jan 2009.

Gelbwaks, N. L., *AFSCM 375-5 as a Methodology for System Engineering*, Systems Science and Cybernetics, IEEE Transactions on 3 (1967), no. 1, 6-10.

Gelosh, D., 2008, *Development and Validation of a Systems Engineering Competency Model*, proceedings of 11th NDIA Systems Engineering Conference.

George, C. S., Jr., *The History of Management Thought*, Prentice-Hall Inc., 1972.

Gharajedaghi, J., *System Thinking: Managing Chaos and Complexity*, Butterworth-Heinemann, Boston, 1999.

Gibbons, P. J., 2001, *A Case Study in the Application of UML and OOA/D in an Information Management Program*, proceedings of The 11th Annual Symposium of the INCOSE.

Glass, R. L., *Building Quality Software*, Prentice Hall, Englewood Cliffs, NJ., 1992.

Goldberg, R. and Assaraf, S., 2010, *System Engineering Process Improvement using the CMMI in Large Space Programs*, proceedings of 13thAnnual NDIA System Engineering Conference 25-28 Oct.

Goldratt, E. M., *Theory of Constraints*, North River Press, 1990.

Goldsmith, R. F., *Discovering Real Business Requirements for Software Project Success*, Artech House Inc., Boston, MA, 2004.

Goleman, D., *Emotional Intelligence*, 1995.

Goode, H. H. and Machol, R. E., *Systems Engineering*, McGraw-Hill, 1959.

Gooding, R. Z., *Systems Engineering: A problem solving approach to improving program performance*, Evaluation and Program Planning 3 (1980), 95-103.

Gordon G. et al., *A Contingency Model for the Design of Problem Solving Research Pro-gram*, Milbank Memorial Fund Quarterly (1974), 184-220.

Gosling, W., *The Design of Engineering Systems*, Wiley, New York, 1962.

Gouillart, F. J. and Kelly, J. N., *Transforming the Organisation*, McGraw-Hill Inc., 1995.

Guo, J., "Incorporating Multidisciplinary Design Optimization into Spacecraft Systems Engineering," *8th Annual Conference on Systems Engineering Research*, Hoboken, NJ., 2010.

Hall, A. D., *A Methodology for Systems Engineering*, D. Van Nostrand Company Inc., Princeton, NJ, 1962.

---, *Metasystems Methodology. a new synthesis and unification*, Pergamon Press, Oxford, England., 1989.

Hall, C. S. and Lindzey, G., *Theories of Personality*, John Wiley & Sons, 1957.

Hambleton, K. G., "Systems Engineering Principles," *Lecture Notes*, University College London, 1999.

Hammer, M. and Champy, J., *Reengineering the Corporation*, HarperCollins, New York, 1993.

Hari, A., Kasser, J. E. and Weiss, M. P., *How lessons learnt from creating requirements for complex systems using QFD led to the evolution of a process for creating quality requirements for complex systems*, Systems Engineering: The Journal of the International Council on Systems Engineering 10 (2007), no. 1, 45-63.

Hari, A., Weiss, M. and Zonnenshain, A., 2004, *ICDM - An Integrated Methodology for the Conceptual Design of New Systems*, proceedings of System Engineering Test and Evaluation Conference SETE 2004.

Harrington, H. J., *Total Improvement Management the next generation in performance improvement*, McGraw-Hill, 1995.

---, *Was W. Edwards Deming wrong?*, Measuring Business Excellence 4 (2000), no. 3, 35 - 41.

Harris, D. D., 2000, *Supporting Human Communication in Network-based Systems Engineering*, proceedings of 2nd European Systems Engineering Conference.

Hart, C. W. L. and Bogan, C. E., *The Baldridge: what it is, how it's won, how to use it to improve quality in your company*, McGraw-Hill, Inc., 1992.

Haskins, C. (Editor), *Systems Engineering Handbook: A Guide for Life Cycle Processes and Activities, Version 3*, The International Council on Systems Engineering, 2006a.

--- (Editor), *Systems Engineering Handbook: A Guide for Life Cycle Processes and Activities, Version 3.1. Revised by K. Forsberg and M. Krueger*, The International Council on Systems Engineering, San Diego, CA., 2006b.

--- (Editor), *Systems Engineering Handbook: A Guide for Life Cycle Processes and Activities, Version 3.2.1. Revised by K. Forsberg and M. Krueger*,

The International Council on Systems Engineering, San Diego, CA., 2011.

Hately, D. J. and Pirbhai, I. A., *Strategies for Real-time systems Specification*, Dorset House Publishing, New York, NY, 1987.

Hauser, J. R. and Clausing, D., *The House of Quality*, Harvard Business Review May-June (1988), 63-65.

Hawley, J. K., *Where's the Q in TQM*, Quality Progress (1995), no. October, 63-64.

Hill, J. D. and Warfield, J. N., *Unified Program Planning*, IEEE Transactions on Systems, Man, and Cybernetics SMC-2 (1972), no. 5, 610-621.

Hiremath, M., *Systems Engineering in Acquisition Strategy: Change Needed,*, INCOSE Insight 11 (2008), no. 5, 32-33.

Hitchins, D. K., *Putting Systems to Work*, John Wiley & Sons, Chichester, England, 1992.

Hitchins, D. K., 1998, *Systems Engineering…In Search of the Elusive Optimum*, proceedings of Fourth Annual Symposium of the INCOSE-UK.

---, World Class Systems Engineering - the five layer Model, 2000, http://www.hitchins.net/5layer.html, accessed on 3 November 2006.

---, *Advanced Systems Thinking, Engineering and Management*, Artech House, 2003.

Hitchins, D. K., *Systems Engineering. A 21st Century Systems Methodology*, John Wiley & Sons Ltd., Chichester, England, 2007.

Holt, J., *UML for Systems Engineering: watching the wheels*, The Institute of Electrical Engineers, 2001.

Honour, E. C. and Valerdi, R., 2006, *Advancing an Ontology for Systems Engineering to Allow Consistent Measurement*, proceedings of Conference on Systems Engineering Research.

Hooks, I., 1993, *Writing Good Requirements*, proceedings of Proceedings of the 3rd NCOSE International Symposium.

---, 1994, *Writing Good Requirements*, proceedings of Proceedings of the 3rd NCOSE International Symposium.

Hopkins, F. W. and Rhoads, R. P., 1998, *Object Oriented Systems Engineering - An Approach*, proceedings of The 8th International INCOSE Symposium.

Howard, R., "The SCADC project," J. E. Kasser (Editor), Adelaide, Australia, 2001.

Hudson, S., Systems Engineering Competencies Framework, INCOSE UK, 2006.

Hull, M. E. C., Jackson, K. and Dick, A. J. J., *Requirements Engineering*, Springer, 2002.

Humphrey, W., *A Discipline for Software Engineering*, Addison-Wesley, Reading, MA, 1995.

Huygens, C., *Treatise on Light*, 1690.

Hyer, S. A., 1997, *An Effective Approach To System Integration: A Comprehensive Checklist*, proceedings of The 7th Annual International Symposium of the INCOSE.

IEEE 610, "IEEE Standard Glossary of Software Engineering Terminology," IEEE, 1990.

IEEE 1220, "Standard 1220 IEEE Standard for Application and Management of the Systems Engineering Process," 1998.

IEEE CCCC, *IEEE Computer Society Computing Curriculum - software engineering --- Public Draft 1 --- (July 17, 2003) Software Engineering Education Knowledge Software*, 2003.

INCOSE, What is systems engineering?, 2000, , (http://www.incose.org/whatis.html, accessed on 9th November 2000.

---, *Systems Engineering Body of Knowledge*, 2002.

---, Certification Examination, 2008, http://www.incose.org/educationcareers/certification/examination.aspx, accessed on 19 February 2012.

INCOSE Fellows, A Consensus of the INCOSE Fellows, 2009, http://www.incose.org/practice/fellowsconsensus.aspx, accessed on 18 March 2009.

INCOSE UK, Systems Engineering Competencies Framework, INCOSE UK, 2010.

Jackson, M. C. and Keys, P., *Towards a System of Systems Methodologies*, Journal of the Operations Research Society 35 (1984), no. 6, 473-486.

Jacobs, S., 1999, *Introducing Measurable Quality Requirements: A Case Study*, proceedings of the IEEE International Symposium on Requirements Engineering.

Jacobson, I., Christerson, M., Jonsson, P. and Ivergarrd, G., *Object-Oriented Software Engineering A Use Case Driven Approach*, Addison Wesley, 1993.

Jain, R. and Verma, D., Proposing a Framework for a Reference Curriculum for a Graduate Program in Systems Engineering, The International Council on Systems Engineering, 2007.

Jansma, P. A. T. and Jones, R. M., 2006, *Advancing the Practice of Systems Engineering at JPL*, proceedings of IEEE Aerospace Conference.

Jarke, M., 1996, *Meta Models for Requirements Engineering*, proceedings of Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop.

JAXA, *Basics of Systems Engineering (draft) , Version 1B*, 2007.

Jenkins, G. M., "The Systems Approach," *Systems Behaviour*, J. Beishon and G. Peters (Editors), Harper and Row, London, 1969, p. 82.

Jenkins, S., 2005, *A Future for Systems Engineering Tools*, proceedings of PDE 2005, The 7th NASA-ESA Workshop on Product Data Exchange (PDE).

Johnson, E. A., "The Executive, the Organisation and Operations Research," *Operations Research for Management, Volume 1.*, J. F. McCloskey and F. N. Trefethen (Editors), The Johns Hopkins Press, Baltimore, 1954.

Johnson, R. A., Kast, F. W. and Rosenzweig, J. E., *The Theory and Management of Systems*, McGraw-Hill, New York, 1963.

Johnson, S. B., *Three Approaches to Big Technology: Operations Research, Systems Engineering, and Project Management*, Technology and Culture (1997), 891-919.

Jones, J. C., *Design Methods: Seeds of human futures*, Wiley-Interscience, 1970.

Jorgensen, R., *The Oxymoron of Use Case Requirements*, INSITE 4 (2001), no. 2.

Jorgensen, R. W., 1998, *Untangling the Twists in Requirements Analysis*, proceedings of the 8th INCOSE International Symposium.

Juran, J. M., *Juran on Planning for Quality*, The Free Press, 1988.

Kanter, R. M., *Attack on Pay*, Harvard Business Review (1987).

Kasser, J. E., 1984, *A Distributed Control System for a Solar Collector Field*, proceedings of Energy 84.

---, *Applying Total Quality Management to Systems Engineering*, Artech House, Boston, 1995.

---, 1996, *Systems Engineering: Myth or Reality*, proceedings of The 6th International Symposium of the INCOSE.

---, "The Determination and Mitigation of Factors Inhibiting the Creation of Strategic Alliances of Small Businesses in the Government Contracting Arena," *The Department of Engineering Management*, The George Washington University, Washington DC, 1997.

---, 1999, *Using Organizational Engineering to Build Defect Free Systems, On Schedule and Within Budget*, proceedings of PICMET.

---, 2000a, *A Framework for Requirements Engineering in a Digital Integrated Environment (FREDIE)*, proceedings of Proceedings of the Systems Engineering, Test and Evaluation Conference.

---, 2000b, *How Collaboration via the World Wide Web Can Provide a Global Perspective and Truly Provide the Student With a World Class Education*, proceedings of Distance Education: An Open Question?

---, 2000c, *A Web Based Asynchronous Virtual Conference: A Case Study*, proceedings of The INCOSE - Mid-Atlantic Regional Conference.

---, "The Cataract Methodology for Systems and Software Acquisition," *SETE 2002*, Sydney Australia, 2002a.

---, 2002b, *Does Object-Oriented System Engineering Eliminate the Need for Requirements?*, proceedings of the 12th International Symposium of the International Council on Systems Engineering.

---, 2002c, *A Prototype Tool for Improving the Wording of Requirements*, proceedings of the 12th International Symposium of the INCOSE.

---, 2002d, *Systems Engineering: An Alternative Management Paradigm*, proceedings of The Systems Engineering Test and Evaluation Conference.

---, *The First Requirements Elucidator Demonstration (FRED) Tool*, Systems Engineering: The Journal of the International Council on Systems Engineering, 7 (2004), no. 3.

---, 2006, *Reducing the cost of doing work by an order of magnitude (by applying systems thinking to systems engineering)*, proceedings of 21st Centre of Excellence Workshop: Challenges for life-based systems development.

---, 2009, *Active Brainstorming: - A systemic and systematic approach for idea generation*, proceedings of the 19th International Symposium of the International Council on Systems Engineering.

---, 2010, *Holistic Thinking and How It Can Produce Innovative Solutions to Difficult Problems*, proceedings of the 7th bi-annual European Systems Engineering Conference (EuSEC).

Kasser, J. E., "An application of Checkland's soft systems methodology in the context of systems thinking," *the 5th Asia-Pacific Conference on Systems Engineering (APCOSE 2011)*, Seoul, Korea, 2011a.

Kasser, J. E., "Applying Holistic Thinking to Improving Your Sex Life," *the Sixth Israeli Conference on Systems Engineering*, Hertzlia, 2011b.

Kasser, J. E., "Complex solutions for complex problems," *Third International Engineering Systems Symposium (CESUN)*, Delft, Holland, 2012.

---, *Holistic Thinking: creating innovative solutions to complex problems*, Createspace, 2013.

Kasser, J. E. and Cook, S. C., 2002, *The Communications Requirements Evaluation & Assessment Prototype (CREAP)*, proceedings of the 12th international symposiunm of the INCOSE.

---, 2003, *Using a Rapid Incremental Solution Construction Approach to Maximise the Completeness and Correctness of a Set of Requirements for a System*, proceedings of Proceedings of the 13th International Symposium of the International Council on Systems Engineering (INCOSE).

Kasser, J. E., Cook, S. C., Scott, W., Clothier, J. and Chen, P., 2002, *Introducing a Next Generation Computer Enhanced Systems Engineering Tool: The Operations Concept Harbinger*, proceedings of SETE 2002.

Kasser, J. E., Frank, M. and Zhao, Y. Y., 2010, *Assessing the competencies of systems engineers*, proceedings of the 7th bi-annual European Systems Engineering Conference (EuSEC).

Kasser, J. E. and Hitchins, D. K., A framework for a systems engineering body of knowledge, 0.6, Report to the Fellows Committee, International Symposium of the International Council on Systems Engineering, 2009.

Kasser, J. E., Hitchins, D. K. and Huynh, T. V., 2009, *Reengineering Systems Engineering*, proceedings of the 3rd Annual Asia-Pacific Conference on Systems Engineering (APCOSE).

Kasser, J. E., John, P., Tipping, K. and Yeoh, L. W., 2008, *Systems engineering a 21st century introductory course on systems engineering: the Seraswati Project*, proceedings of the 2nd Asia Pacific Conference on Systems Engineering.

Kasser, J. E., Kaffle, S. and Saha, P., 2007, *Applying FRAT to improve systems engineering courseware: Project Review*, proceedings of SEEC Research Group, SESA-South Australia and INCOSE-Australia joint meeting.

Kasser, J. E. and Mackley, T., 2008, *Applying systems thinking and aligning it to systems engineering*, proceedings of the 18th INCOSE International Symposium.

Kasser, J. E. and Mirchandani, C. J., 2005, *The MSOCC Data Switch Replacement: A Case Study in Elicitating and Elucidating Requirements*, proceedings of The 15th International Symposium of the International Council on Systems Engineering (INCOSE).

Kasser, J. E. and Schermerhorn, R., 1994a, *Determining Metrics for Systems Engineering*, proceedings of The 4th Annual International Symposium of the NCOSE.

---, 1994b, *Gaining the Competitive Edge through Effective Systems Engineering*, proceedings of The 4th Annual International Symposium of the NCOSE.

Kasser, J. E., Scott, W., Tran, X.-L. and Nesterov, S., 2006, *A Proposed Research Programme for Determining a Metric for a Good Requirement*, proceedings of Conference on Systems Engineering Research.

Kasser, J. E., Sitnikova, E., Tran, X.-L. and Yates, G., 2005, *Optimising the Content and Delivery of Postgraduate Education in Engineering Management for Government and Industry*, proceedings of the International Engineering Management Conference (IEMC).

Kasser, J. E., Tran, X.-L. and Matisons, S., 2003, *Prototype Educational Tools for Systems and Software (PETS) Engineering*, proceedings of Proceedings of the AAEE Conference.

Kast, F. E. and Rosenzweig, J. E., *Organization and Management A Systems and Contingency Approach*, McGraw-Hill Book Company, 1979.

Kemp, L. L., Nidiffer, K. E., Rose, L. C., Small, R. and Stankowsky, M., *Knowledge Management: Insights from the Trenches*, IEEE Software (2001), no. November/December, 66-68.

Kendall, K. E. and Kendall, J. E., *Systems Analysis and Design*, Prentice Hall, Upper Saddle River, NJ, 1997.

Kezsbom, D. S., Schilling, D. L. and Edward, K. A., *Dynamic Project Management. A practical guide for managers and engineers*, John Wiley & Sons, New York, 1989.

Kirton, M. J., *Adaptors and Innovators: Styles of Creativity and Problem Solving*, Routledge, London, 1994.

Kline, S. J., *Conceptual Foundations for Multidisciplinary Thinking*, Stanford University Press, Stanford, 1995.

Kossiakoff, A. and Sweet, W. N., *Systems Engineering: Principles and practice*, John Wiley & Sons Inc., 2003.

Kossmann, M., Odeh, M., Gillies, A. and Ingamells, C., 2007, *'Tour d'horizon' in Requirements Engineering - Areas left for exploration*, proceedings of the 17th International Symposium of the INCOSE.

Kotonya, G. and Summerville, I., *Requirements Engineering processes and techniques*, John Wiley & Sons, Chichester, 2000.

Kuhn, T. S., *The Structure of Scientific Revolutions*, The University of Chicago Press, 1970.

Lagakos, V., Kaisar, E. I. and Austin, M., 2001, *Object Modeling for the Management of Narrow Passageways in Transportation Systems*, proceedings of The 11th Annual Symposium of the INCOSE.

Lander, S. E., *Issues in Multiagent Design Systems*, IEEE Expert (1997), no. 12 Issue: 2, March-April.

LaPlue, L., Garcia, R. and Rhodes, R., 1995, *A Formal Method for Rigorous Requirements Definition*, proceedings of Fifth Annual Symposium of the NCOSE.

LaRocca, M., 1999, *Career and Competency Pathing: The Competency Modeling Approach*, proceedings of Linkage Conference on Emotional Intelligence.

Lawler III, E., *Motivation in Work Organizations*, Brooks/Cole Publishing Company, 1973.

Lee, J. Y. and Park, Y. W., 2004, *Requirement Architecture Framework (RAF)*, proceedings of the 14th International Symposium of the INCOSE.

Leveson, N., *The Role of Software in Spacecraft Accidents*, AIAA Journal of Spacecraft and Rockets 41 (2004), no. 4.

Lewicki, R. J. and Litterer, J. A., *Negotiation.*, IRWIN, 1985.

Lewis, R. O., *Independent Verification & Validation*, John Wiley & Sons, 1992.

Love, T., 2003, *A Fork in the Road: Systems and Design*, proceedings of 9th ANZSYS Conference.

Luzatto, M. C., *The Way of God*, Feldheim Publishers, 1999, New York and Jerusalem, Israel, circa 1735.

Lykins, H., Friedenthal, S. and Meilich, A., 2000, *Adapting UML for an Object Oriented Systems Engineering Method (OOSEM)*, proceedings of The 10th Annual Symposium of the INCOSE.

M'Pherson, P., *Systems engineering: A proposed definition*, IEE Proc 133 (1986), no. September.

Mackey, W. F. and Mackey, W. F., Jr., 1994, *A Systems Engineering Approach to Highway Design*, proceedings of 4th Annual International Symposium of the NCOSE.

Maier, M. K. and Rechtin, E., *The Art of Systems Architecting*, CRC Press, 2000.

Maimonides, M., *The Guide for the Perplexed translated by M Friedlander*, Dover Publications, circa 1200.

Mar, B., Commentary on the Consensus of INCOSE Fellows, 2009a, http://www.incose.org/practice/fellowsconsensus.aspx, accessed on 16 April 2012.

Mar, B., Commentary on the Consensus of INCOSE Fellows, INCOSE, 2009b, http://www.incose.org/practice/fellowsconsensus.aspx, accessed on 8 November 2009.

Mar, B. and Morais, B., 2002, *FRAT A Basic Framework for Systems Engineering*, proceedings of 12th annual International Symposium of the International Council on Systems Engineering.

Mar, B. W., *Systems Engineering Basics*, Systems Engineering: The Journal of INCOSE 1 (1994), no. 1, 7-15.

Marcotty, M., *Software Implementation*, Prentice Hall, 1991.

Martin, J. N., 1994, *The PMTE Paradigm: Exploring the Relationship Between Systems Engineering Processes and Tools*, proceedings of 4th Annual International Symposium of the National Council on Systems Engineering, INCOSE.

Martin, J. N., 1996, *On The Nature of Integration: An Essential Task in the Engineering of Systems*, proceedings of 6th International Symposium of the International Council on Systems Engineering (INCOSE).

Martin, J. N., *Systems Engineering Guidebook: A process for developing systems and products*, CRC Press, 1997.

Martin, N. G., 2005, *Work Practice in Research: A Case Study*, proceedings of the 14th International Symposium of the INCOSE.

Maslow, A. H., *A Theory of Human Motivation*, Harper & Row, 1954.

---, *The Psychology of Science*, Harper and Row, 1966.

---, *Toward a Psychology of Being*, Van Nostrand, 1968.

---, *Motivation and Personality*, Harper & Row, 1970.

Mason, G. A., Sherwood, W. B., McGrath, W. E., Silva, F. G., Rutter, C. K. and Spence, J. P., 1999, *Application of System Engineering Principles in the Development of the Advanced Photo System*, proceedings of the 9th INCOSE International Symposium.

Matchett, E. and Briggs, A. H., "Practical Design Based on Method," *The Design Method*, S. Gregory (Editor), Butterworths, London, 1966.

McCloskey, J. F., "Case Histories in Operations Research," *Operations Research for Management, Volume 1.*, J. F. McCloskey and F. N. Trefethen (Editors), The Johns Hopkins Press, Baltimore, 1954.

McConnell, G. R., 2002, *Emergence : A Partial History of Systems Thinking*, proceedings of the 12th international symposiunm of the INCOSE.

McGregor, D., *The Human Side of Enterprise*, McGraw-Hill, 1960.

McNaugher, T. L., Risks and Rushing: The Causes and Costs of Production Concurrency", Statement prepared for the House Committee on Government Reform, Subcommittee on National Security, Veterans Affairs, and International Relations, hearings entitled Joint Strike Fighter (JSF) Acquisition Reform: Will It Fly?, 2000, http://www.fas.org/man/congress/2000/000510-mcnaugher_may_10.htm, accessed on April 5, 2001.

Meilich, A. and Rickels, M., 1999, *An Application of Object Oriented Systems Engineering (OOSE) To an Army Command and Control System: A New Approach to Integration of System and Software Requirements and Design*, proceedings of the 9th International INCOSE Symposium.

Memory Jogger, "Memory Jogger," GOAL/QPC, Mathuen, MA, 1985.

Menrad, R. and Larson, W., 2008, *Development of a NASA integrated technical workforce career development model entitled: Requisite occupation competencies and knowledge --the ROCK*, proceedings of the 59th International Astronautical Congress (IAC).

Menzes, T., Easterbrook, S., Nuseibeh, B. and Waugh, S., 1999, *An empirical investigation of multiple viewpoints reasoning in requirements engineering*, proceedings of IEEE International Symposium on Requirements Engineering.

Mesarovic, M. D. (Editor), *Views on General Systems Theory*, Wiley, New York, 1964.

Metzger, L. S. and Bender, L. R., *MITRE Systems Engineering (SE) Competency Model Version 1.13E*, The MITRE Corporation, 2007.

Metzger, P. W. and Boddie, J., *Managing a Programming Project Processes and People*, Prentice Hall PTR, Upper Saddle River, NJ, 1996.

Microsoft, Transforming Education: Assessing and Teaching 21st Century Skills, 2008a, http://download.microsoft.com/download/6/E/9/6E9A7CA7-0DC4-4823-993E-A54D18C19F2E/Transformative%20Assessment.pdf, accessed on 20 March 2009.

---, Transforming Education: Assessing and Teaching 21st Century Skills,, 2008b, http://download.microsoft.com/download/6/E/9/6E9A7CA7-0DC4-4823-993E-A54D18C19F2E/Transformative%20Assessment.pdf, accessed on 20 March 2009.

MIL-HDBK-61A, *Military Handbook: Configuration Management Guidance*, Department of Defense, 2001.

MIL-STD-490A, *Specification Practices*, United States Department of Defense, 1985.

MIL-STD-499, *Mil-STD-499 Systems Engineering Management*, United States Department of Defense (USAF), 1969.

MIL-STD-499A, *Mil-STD-499A Engineering Management*, United States Department of Defense (USAF), 1974.

MIL-STD-499B, *Mil-STD-499B Systems Engineering*, United States Department of Defense, 1992.

---, Draft MIL-STD-499B Systems Engineering, United States Department of Defense, 1993.

MIL-STD-2167A, *Defense System Software*, United States Department of Defense, 1998.

Miles, R. F., *Systems Concepts*, Wiley, New York, 1973.

Miller, G., *The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information.*, The Psychological Review 63 (1956), 81-97.

Milne, A. A., *The Christopher Robin Story Book*, Methuen Children's Books, London, 1929.

Moore, J. W., *Software Engineering Standards A User's Road Map*, IEEE Computer Society, Los Alamitos, CA., 1998.

Morton, J. A., *Integration of Systems Engineering with Component Development*, Electrical Manufacturing 64 (1959), no. August 1959, 85-92.

Mueller, D. M., *Capability Systems Life Cycle Management Guide, November 1, 2001*, Department of Defence, Australian Defence Organisation, Canberra, ACT, 2001.

Murray, H. A., *Explorations in Personality*, Oxford University Press, 1938.

Myers, G., *The Art of Software Testing*, Wiley, 1979.

NASA, *NASA Systems Engineering Handbook - draft September 1992*, 1992a.

---, *NASA Systems Engineering Handbook, draft, September 1992*, 1992b.

---, NASA's systems engineering competencies, 2010, http://www.nasa.gov/pdf/303747main_Systems_Engineering_Competencies.pdf, accessed on 29 August 2011.

NDIA E&T, December 8, 2010 Meeting: Education & Training Committee Report, NDIA, 2010.

Newland, K. E., 1998, *Risks Associated with the Application of Systems Engineering*, proceedings of Fourth Annual Symposium of the INCOSE-UK.

Newton, I., *Hypothesis of Light*, 1675.

O'Reilly, J., 2004, *Message from the President of the IEE*, proceedings of The International Engineering Management Conference.

O'Toole, P., 2004, *Do's and Don'ts of Process Improvement*, proceedings of 2005 U.S. Census Bureau SEPG Conference.

OASIG, The performance of information technology and the role of human and organizational factors. Report to the Economic and Social Research Council, UK, 1996, http://www.shef.uc.uk/~iwp/publications/reports/itperf.html, accessed on January 16, 2002.

Ogren, I., *Comment on the use case Requirements Oxymoron*, INSITE 4 (2001), no. 3.

OPM, Policies and instructions, U.S. Office of Personnel Management, 2009, http://www.opm.gov/qualifications/policy/index.asp, accessed on 29 October 2009.

OVAE, Problem-Solving Process, Office of Vocational and Adult Education (OVAE), US Department of Education, 2005, http://www.c-pal.net/course/module3/pdf/Week3_Lesson21.pdf, accessed on 11 Jan 2009.

Palmer, S. R. and Felsing, J. M., *A Practical Guide to Feature - Driven Development*, Prentice Hall, 2002.

Paul, R. and Elder, L., *Critical Thinking: learn the tools the best thinkers use - concise ed.*, Pearson Prentice Hall, 2006.

Pearson, W. D., 2000, *The Role of Test and Evaluation: A United States Air Force Perspective*, proceedings of Fourth Test and Evaluation International Aerospace Forum.

Pennell, L. W. and Knight, F. L., Draft MIL-STD 499C Systems Engineering, The Aerospace Corporation, 2005.

Perry, W. E., *Effective Methods for Software Testing*, John Wiley & Sons, 2000.

Peter, L. J. and Hull, R., *The Peter Principle*, William Morrow & Company, Inc, New York, 1969.

Peters, J. F. and Pedrycz, W., *Software Engineering An Engineering Approach*, John Wiley & Sons, Inc., New York, 2000.

Peters, T., *Thriving on Chaos: Handbook for a Management Revolution*, Harper and Row, 1987.

Peters, T. and Austin, N., *A Passion for Excellence*, Warner Books, 1985.

Peters, T. J. and Waterman, H. R., *In Search of EXCELLENCE*, Harper and Row, 1982.

Pfleeger, S. L., *Software Engineering Theory and Practice*, Prentice Hall, New Jersey, 1998.

Phillips, D., *The Software Project Manager's Handbook Principles that Work at Work*, IEEE Computer Society, Los Alamitos, CA., 1998.

Pike, J., M247 Sergeant York DIVAD, 1999, http://www.fas.org/man/dod-101/sys/land/m247.htm,, accessed on April 5, 2001.

---, Joint Strike Fighter (JSF), 2000, http://www.fas.org/man/dod-101/sys/ac/jsf.htm, accessed on April 5, 2001.

PMI, *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*, Project Management Institute, 2004.

Powell, R. A. and Buede, D., 2006, *Innovative Systems Engineering: A Creative System Development Approach*, proceedings of the 16th International Symposium of the International Council on Systems Engineering (INCOSE).

Pressman, R. S., *Software Engineering A Practitioner's Approach*, McGraw-Hill, 2005.

Priest, J. W. and Sánchez, J. M., *Product Development and Design for Manufacturing*, Marcel Dekker, 2001.

Prieto-Díaz, R. (Editor), *Classification of Reusable Models published in Software Reusability, 1987*, ACM Press, 1987.

Rechtin, E., *Systems Architecting, Creating & Building Complex Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1991.

Reid, P. P., Compton, W. D., Grossman, J. H. and Fanjiang, G. (Editors), *Building a Better Delivery System: A New Engineering/Health Care Partnership*, The National Academies Press, Washington, D.C., 2005.

Rhodes, D. H., 2002, *Systems Engineering on the Dark Side of the Moon*, proceedings of the 12th International Symposium of the INCOSE.

Richmond, B., *Systems thinking: critical thinking skills for the 1990s and beyond*, System Dynamics Review 9 (1993), no. 2, 113-133.

Rittel, H. W. and Webber, M. M., *Dilemmas in a General Theory of Planning*, Policy Sciences 4 (1973), 155-169.

Robson, P. G., 2001, *Systems Engineering - In Retrospect and in Prospect*, proceedings of Fifth Annual Symposium of the INCOSE-UK.

Rodgers, T. J., Taylor, W. and Foreman, R., *No-excuses Management*, Doubleday, 1993.

Roe, C. L., 1995, *The Role of the Project Manager in Systems Engineering*, proceedings of The 5th Annual International Symposium of the NCOSE.

Rook, P., *Controlling software projects*, Software Engineering Journal 1 (1986), no. 1, 7-16.

Roy, R. H., "The Development and Future of Operations Research and Systems Engineering," *Operations Research and Systems Engineering*, C. D. Flagle, W. H. Huggins and R. H. Roy (Editors), Johns Hopkins Press, Baltimore, 1960.

Royce, W. W., 1970, *Managing the Development of Large Software Systems*, proceedings of IEEE WESCON.

Saaty, T., *The Analytic Hierarchy Process*, McGraw Hill, New York, NY, 1980.

Saaty, T., *Decision Making for Leaders*, RWS Publications, Pittsburgh, PA, 1990.

Sage, A. P., *Systems Engineering*, John Wiley & Sons, Ltd., 1992.

Saxe, J. G., *The Poems of John Godfrey Saxe, Complete edition*, James R. Osgood and Company, Boston, 1873.

SBA, Small Business Research Summary, U.S. Small Business Administration, 1982.

Schach, S., *Object-Oriented and Classical Software Engineering*, McGraw Hill, 2002.

Scott, W., Kasser, J. E. and Tran, X.-L., 2006, *Improving the Structure and Content of the Requirement Statement*, proceedings of The 16th International Symposium of the International Council on Systems Engineering (INCOSE).

Scuderi, P. V., What is a System, 2004, http://www.wsu.edu:8001/vwsu/gened/learn-modules/top_system/resources/system.html, accessed on July 1.

Selby, R. W., 2006, *Enabling Measurement-Driven System Development by Analyzing Testing Strategy Tradeoffs*, proceedings of The 16th International Symposium of the INCOSE.

Senge, P. M., *The Fifth Discipline: The Art & Practice of the Learning Organization*, Doubleday, New York, 1990.

Shaw, G. B., *England and America: Contrasts," a conversation between Bernard Shaw and Archibald Henderson*, Reader's Digest (1925).

Sheard, S. A., 1996, *Twelve Systems Engineering Roles*, proceedings of The 6th Annual International Symposium of the NCOSE.

---, 2003, *Process Implementation*, proceedings of The 13th Annual Symposium of the INCOSE.

Shenhar, A. J. and Bonen, Z., *The New Taxonomy of Systems: Toward an Adaptive Systems Engineering Framework*, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 27 (1997), no. 2, 137 - 145.

Shinner, S. M., *A Guide to Systems Engineering Management*, Lexington Books, Lexington Massachusetts, 1976.

Shlaer, S. and Mellor, S. J., *Object Oriented Systems Analysis*, Yourdon Press, 1988.

Shumate, K. and Keller, M., *Software Specification and Design A Disciplined Approach for Real-Time Systems*, John Wiley & Sons Inc., 1992.

Sillitto, H., 2008, *Systems and System-of-Systems Architecting*, proceedings of DSTL Systems Skills Symposium, The Defence Science and Technology Laboratory (Dstl).

Singh, R. (Editor), *ISO 12207 International Standard on Software Lifecycle Processes*, 1995.

Sommerville, I., *Software Engineering*, Addison-Wesley, 1998.

Squires, A., "Qualifying Exam: Systems Thinking and K12 Education," Stevens Institute of Technology, Hoboken, NJ., 2007.

Squires, A., Wade, J., Dominick, P. and Gelosh, D., 2011, *Building a competency taxonomy to guide experience acceleration of lead program systems engineers*, proceedings of the 9th conference on systems engineering research (CSER 2011).

STDADS, *ST DADS Requirements Analysis Document (FAC STR-22), Rev. C, August 1992, as modified by the following CCR's:- 139, 146, 147C, 150 and 151B*, NASA/Ford AeroSpace, Greenbelt, MD, 1992.

Stevens, K. L., 2003, *DMSMS Tutorial 1 - Session 1*, proceedings of The 2003 DoD Diminishing Manufacturing Sources and Materials Shortages (DMSMS) Conference.

Sutcliffe, A., Galliers, J. and Minocha, S., 1999, *Human Errors and System Requirements*, proceedings of IEEE International Symposium on Requirements Engineering.

Swarz, R. S. and DeRosa, J. K., 2006, *A Framework for Enterprise Systems Engineering Processes*, proceedings of International Conference on Computer Science, Engineering and Applications (ICSSEA).

Szyperski, C., *Component Software*, Addison Wesley, 1997.

Taylor, F. W., *The Principles of Scientific Management*, Harper & Brothers Publishers, 1911.

Thomas, J. A., 2011, *Energy-Environment-Economic System of Systems*, proceedings of 5th Asia-Pacific Conference on Systems Engineering.

Thome, B. (Editor), *Systems Engineering Principles and Practice of Computer-based Systems Engineering*, John Wiley & Sons, 1993.

Tittle, P., *Critical Thinking: An appeal to reason*, Routledge, 2011.

Todaro, R. C., "Lecture Handout, ENEE 648R," University of Maryland University College, 1988.

Tran, X.-L. and Kasser, J. E., 2005, *Improving the recognition and correction of poorly written requirements*, proceedings of The Systems Engineering Test and Evaluation (SETE) Conference.

---, 2007, *Systems Engineering Tools for Australian Small and Medium Enterprises*, proceedings of the Asia Pacific Systems Engineering Conference.

UML, OMG Unified Modeling Language Specification Version 1.3, OMG, 1999.

---, Unified Modeling Language: Superstructure, OMG, 2005.

UNiSA, "Module 7: Introduction to Traditional Systems Engineering and Life Cycle Modelling," *Systems Engineering for Complex Problem Solving*, University of South Australia, Adelaide, Australia, 2006.

Van Gaasbeek, J. R. and Martin, J. N., 2001, *Getting to Requirements: The W5H Challenge*, proceedings of The 11th Annual Symposium of the INCOSE.

Van Vliet, H., *Software Engineering Principles and Practice*, John Wiley & Sons Ltd., 2000.

Verma, D., Larson, W. and Bromley, L., 2008, *Space systems engineering: An academic program reflecting collaboration between government, industry and academia (open academic model)*, proceedings of hte 59th International Astronautical Congress (IAC).

Von_Knethen, A., Paech, B., Kiedaisch, F. and Houdek, F., 2002, *Systematic Requirements Recycling through Abstraction and Traceability*, proceedings of IEEE Joint International Conference on Requirements Engineering.

VOYAGES, Unfinished Voyages, A follow up to the CHAOS Report, 1996, http://www.pm2go.com/sample_research/unfinished_voyages_1.asp, accessed on January 21, 2002.

Ward, P. T. and Mellor, S. J., *Structured Development for Real-Time Systems*, Yourdon Press, 1985.

Wasson, C. S., *System Analysis, Design, and Development concepts, principles and practices*, Wiley-Interscience, Hoboken, New Jersey, 2006.

Watts, J. G. and Mar, B. W., 1997, *Important Skills and Knowledge to Include in Corporate Systems Engineering Training Programs*, proceedings of The 7th Annual International Symposium of the INCOSE.

Webster, Merriam-Webster Online Dictionary, 2004, http://www.webster.com, accessed on 12 January 2004.

Weinberg, G. M., *The Psychology of Computer Programming, Silver Anniversary Edition*, Dorset House Publishing, 1998.

Welby, S., 2010, *DoD Systems Engineering Update*, proceedings of 36th Air Armament Symposium.

West, M., 2010, *Improving Performance Through Process Improvement*, proceedings of the 10th Annual CMMI Technology Conference and User Group.

Westerman, H. R., *Systems Engineering Principles and Practice*, Artech House, Boston, 2001.

Wheatcraft, L. S., 2011, *Triple Your Chance of Project Success Risk and Requirements*, proceedings of the 21st Annual Symposium of the INCOSE.

Williams, T. J., *Systems Engineering for the Process Industries*, McGraw-Hill, 1961.

Wilson, W. E., *Concepts of Engineering System Design*, McGraw-Hill Book Company, 1965.

Wolcott, S. K. and Gray, C. J., Assessing and Developing Critical Thinking Skills, 2003, http://www.wolcottlynch.com/Downloadable_Files/IUPUI%20Handout_031029.pdf, accessed on

Wood, J. and Silver, D., *Joint Application Development*, John Wiley and Sons Inc., 1995.

Woolfolk, A. E., "Chapter 7 Cognitive views of learning," *Educational Psychology*, Allyn and Bacon, Boston, 1998, pp. 244-283.

Wymore, A. W., *Systems Engineering Methodology for Interdisciplinary Teams*, John Wiley and Sons, 1976.

---, *Model-Based Systems Engineering*, CRC Press, Boca Raton, 1993.

---, *Model-Based Systems Engineering*, Systems Engineering: The Journal of INCOSE 1 (1994), no. 1, 83-92.

Wynne, M. W., "Policy for Systems Engineering in DoD," Memo by Undersecretary of Defense for Acquisition, Technology, and Logistics, Washington, DC, 2004.

Wynne, M. W. and Schaeffer, M. D., *Revitalization of Systems Engineering in DoD*, Defense AT&L (2005), 14-17.

Yen, D. H., The Blind Men and the Elephant, 2008, http://www.noogenesis.com/pineapple/blind_men_elephant.html, accessed on 26 October 2010.

Yourdon, E., *Decline & Fall of the American Programmer*, Yourdon Press, 1993.

Yourdon, E. and Argila, C., *Object Oriented Analysis and Design*, Yourdon Press, 1993.

33 Index