DRAFT

Eight deadly defects in systems engineering and how to fix them

Joseph E. Kasser DSc, CEng, CM
Systems Engineering and Evaluation Centre
University of South Australia
Mawson Lakes, SA. 5095
Joseph.kasser@incose.org

Abstract. Any organization desirous to adopt or improve systems engineering needs to be aware that research into the nature of systems engineering has identified a number of defects in the current systems engineering paradigm. This paper discusses eight of these defects and ways to fix or compensate for them.

Eight deadly defects in systems engineering

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:

- [Reid ;Compton ;Grossman and Fanjiang, 2005] reported that the health care industry in the United States of America 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.
- The United States [of America] Department of Defense 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 paper are:

1. The selection of independent alternative solutions;
2. The V Model lacks a feed forward or prevention component;
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.

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.

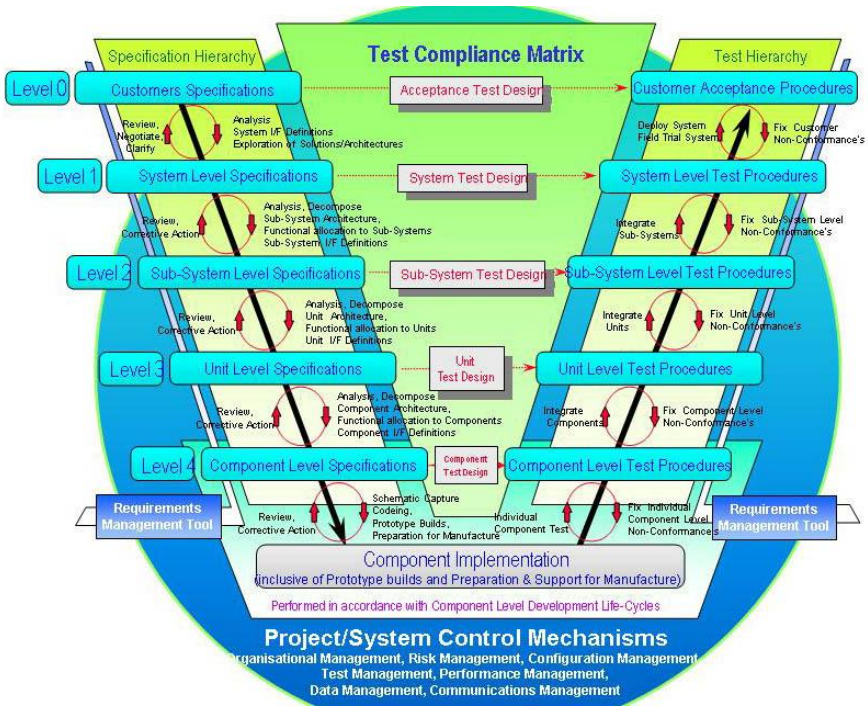

Figure 1 The V Diagram [Miller, 1998]

2. The V Model lacks a feed forward or prevention component

The V model is often described as a depiction of the systems engineering process. A typical example taken from [Miller, 1998] is shown in Figure 1. Practitioners tend to forget that it is only one view of system engineering; a view that relates development to test and evaluation (T&E) at the various phases of the System Development Life Cycle (SDLC) while abstracting out all other information. However, within the relationship between development and T&E there seems to be no place in the V-Model for the 'prevention of defects'. While the development team implements the system, the test team is busy planning the tests. The

definition of a successful test is one that finds defects[1] [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, 1986] page 29) wrote "*Quality comes not from inspection, but from improvement of the production process*". 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 university of Maryland University Collage and the University of South Australia have sensitised students to the problems caused by poorly written requirements [Kasser ;Tran and Matisons, 2003]. The development and test teams need to work interdependently not independently [Kasser, 1995].

The V diagram needs to be modified to explicitly add 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 ;Aquilano and Jacobs, 1998] page 155).

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 suggested process based on experience is shown in Figure 2. The contribution of this process is two important elements that are not generally performed, namely:

- 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 have to be identified. The process architect asks the following questions:

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 database. 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 standards 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!

---

[1] As opposed to the goal of the system development team to produce a defect free system.

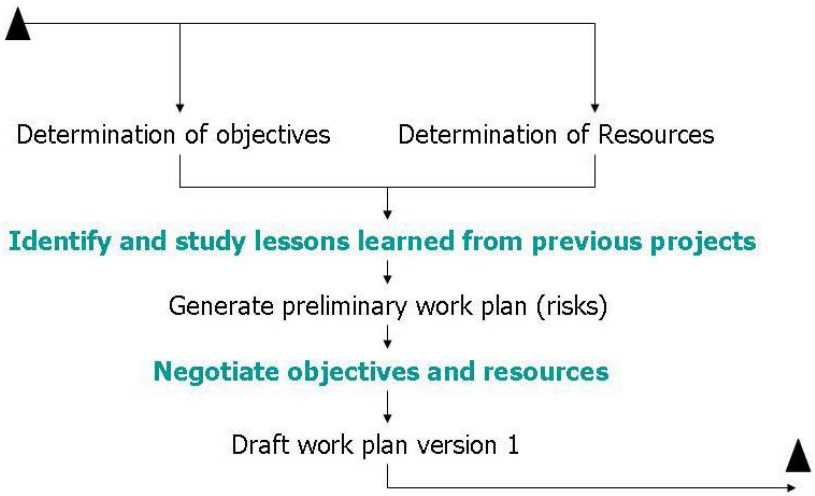

Figure 2 Process for starting a task

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 Work Breakdown Structure (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[2] [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 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.

---

[2]There weren't any problems, so obviously cost and schedule were underestimated.

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, as discussed in [Kasser, 2002] the waterfall model works very well when the requirements are known and don't change during the development time. The spiral model [Boehm, 1988] is basically several iterations of the waterfall model in series 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. The Cataract model discussed in [Kasser, 2002] is not iterative; 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.

5. Unanswered and unasked questions

Unanswered questions. As discussed in [Kasser, 1997] and [Kasser and Williams, 1998], there are two critical questions that cannot be answered accurately in today's systems engineering development paradigm at anytime during the SDLC. 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.

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 discussed in [Kasser, 2003] 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 [Shenhar and Bonen, 1997] associated with the project. 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. Early research into developing ways of presenting decision makers with the information to allow them to easily pose and answer these questions was documented in [Tran and Kasser, 2005].

6. The lack of a metric for the goodness of requirements

According to [Buren and Cook, 1998] *"it has been known since as early as the 1950s that addressing requirements issues improves the chance of systems development success."* 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. However, recognition that the current paradigm produces poorly written requirements has been documented at least as early as [Hooks, 1993] and various approaches have been since proposed to alleviate the situation without much success. For example:

- [Jacobs, 1999] 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. 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, 2004] states that the process of "*defining business requirements is the most important and poorest performed part of system development*".

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 life cycle towards a central focus of change management in system-intensive organizations [Jarke, 1996]. For example:

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

1. However, there is no universally accepted metric for the goodness of requirements either individually or as a set in a specification as discussed in [Kasser ;Scott ;Tran and Nesterov, 2006]. We need to develop one. [Kasser, et al., 2006] proposed a number of ways of developing such metrics.

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 according to [Farnham, 1920] page 20), the problem the executive had was "*to secure at all times, live and accurate data concerning the exact conditions of the business*". 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, [Beer, 1972] page 244) 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. 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 [Kasser, 2000]. 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. A representation of the system taken from [Checkland and Holwell, 1998] page 135) is shown in Figure 3. The system contains the radar sites, the Observer Corps, the communications links and the various headquarters and operations rooms. According to Checkland and Holwell 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 the two 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:

1. 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 have been deployed as part of the installations.
2. The operations rooms were co-sited with airfields. 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. An alternative set of tactics pointed out by [Bungay,

2000] would have exploited these defects to cause much more and serious damage to the RAF infrastructure.

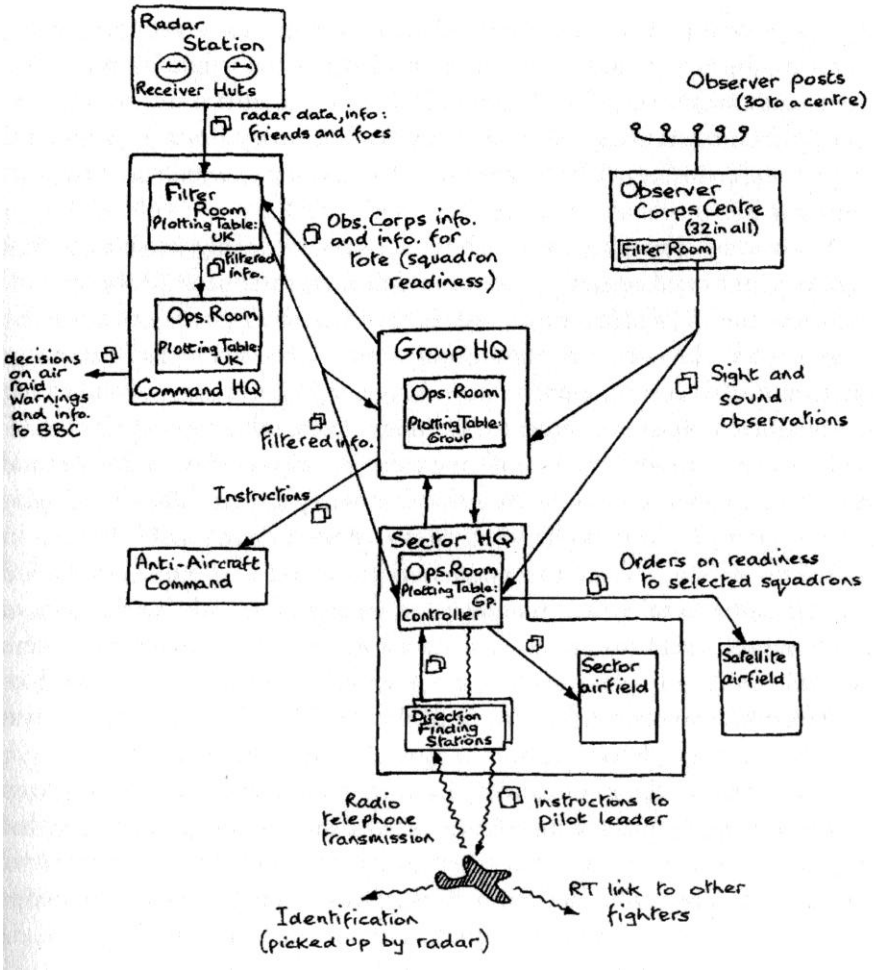

Figure 3 Battle of Britain Command and Control system [Checkland and Holwell, 1998] page 135

Modern systems engineering needs to be able to develop systems in the same manner as the RAF developed this system.

8. The need to focus on people as well as process

Systems engineering is perceived as a process [MIL-STD-499B, 1992] and there is a major focus on process standards and Capability Maturity Models (CMM). The contribution of effective people and the difference they can make is generally overlooked. 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 mongst 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[3], it was the person who made the difference[4]. One was an administrator, the other a leader!

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 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).

[Hall, 1962] pages 16-18) provided the following specifications or traits for an "Ideal Systems Engineer". The specifications are grouped in several areas as:

- 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. 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.

The literature is full of advice as to how to make projects succeed; typical examples are [Harrington, 1995, Peters, 1987, Peters and Austin, 1985, Peters and Waterman, 1982, Rodgers ;Taylor and Foreman, 1993] 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.

Consider the CMM. 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 4. When you get to Level 5 which way do you look, back the way you came, or further up the mountain? If you look back, you see that Level 1 is categorised by having success achieved by heroes. 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

---

[3] 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.

[4] 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?

improvement. We need to look forward up the mountain and explore what lies beyond the base camp at Level 5 to continue the journey further up the process improvement mountain. Thus CMM Level 5 is only a start.

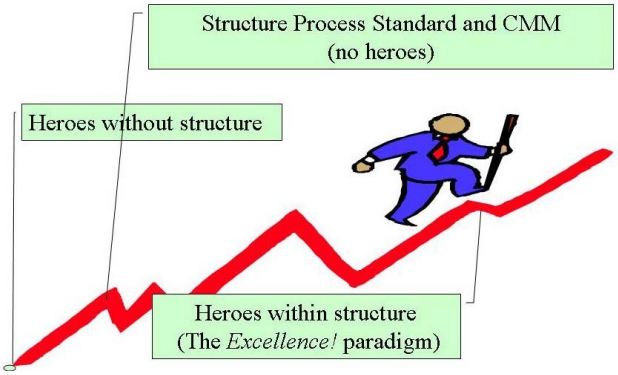

Figure 4 The process improvement mountain

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.

Summary

This paper has identified and discussed eight defects in today's systems engineering paradigm. The paper 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.

Conclusion

Adopting systems engineering without fixing these defects will not realise the potential of systems engineering. We need to train more effective systems engineers.

Biography

Dr. Joseph Kasser has been a practising systems engineer for more than 35 years. He is a Fellow of the INCOSE and the author of "Applying Total Quality Management to Systems Engineering" and many INCOSE symposia papers. He holds a Doctor of Science in Engineering Management from The George Washington University, and is a Certified Manager. He is Deputy Director and Associate Research Professor at the Systems Engineering and Evaluation Centre at the University of South Australia. He teaches online and in the classroom as well as performing research into the nature of systems engineering and the properties of object-oriented requirements. He is a recipient of NASA's Manned Space Flight Awareness Award (Silver Snoopy) for quality and technical excellence for performing and directing systems engineering and many other awards and commendations.

References

S. Beer, Brain of the Firm - A development in Management Cybernetics, Herder and Herder, NY, 1972.

C. Bentley, PRINCE 2 A Practical Handbook, Oxford: Butterworth Heinemann, 1997.
B. Blanchard and W. Fabrycky, Systems Engineering and Analysis, Pearson Prentice Hall, Upper Saddle River, NJ, 2006.
B. Boehm, A Spiral Model of Software Development and Enhancement, IEEE Computer, (1988), Vol No May,
S. Bungay, The Most Dangerous Enemy, Aurum Press, London, England, 2000.
J. V. Buren and D. A. Cook, Experiences in the Adoption of Requirements Engineering Technologies, Crosstalk - The Journal of Defense Software Engineering, (1998), Vol No December,
R. B. Chase, N. J. Aquilano and F. R. Jacobs, Productions and Operations Management, Irwin McGraw-Hill, 1998.
R. B. Chase and D. M. Stewart, Make Your Service Fail-Safe, Sloan Management Review, (1994), Vol 35, No 3, 35-44.
P. Checkland and S. Holwell, Information, Systems and Information Systems: making sense of the field, John Wiley & Sons, 1998.
W. E. Deming, Out of the Crisis, MIT Center for Advanced Engineering Study, 1986.
M. Dorfman and R. H. Thayer, System and Software Requirements Engineering, IEEE Computer Society Press, 1990.
D. T. Farnham, Executive Statistical Control, Industrial Extension Institute, New York, 1920.
R. L. Glass, Building Quality Software, Prentice Hall, Englewood Cliffs, NJ., 1992.
R. F. Goldsmith, Discovering Real Business Requirements for Software Project Success, Artech House Inc., Boston, MA, 2004.
H. H. Goode and R. E. Machol, Systems Engineering, McGraw-Hill, 1959.
A. D. Hall, A Methodology for Systems Engineering, D. Van Nostrand Company Inc., Princeton, NJ, 1962.
M. Hammer and J. Champy, Reengineering the Corporation, HarperCollins, New York, 1993.
H. J. Harrington, Total Improvement Management the next generation in performance improvement, McGraw-Hill, 1995.
E. C. Honour and R. Valerdi, Advancing an Ontology for Systems Engineering to Allow Consistent Measurement, Conference on Systems Engineering Research, Los Angeles, CA., 2006.
I. Hooks, Writing Good Requirements, Proceedings of the 3rd NCOSE International Symposium, 1993.
S. Jacobs, Introducing Measurable Quality Requirements: A Case Study, the IEEE International Symposium on Requirements Engineering, Limerick, Ireland, 1999.
M. Jarke, Meta Models for Requirements Engineering, Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop, Banf, Alberta, Canada, 1996.
J. E. Kasser, Applying Total Quality Management to Systems Engineering, Artech House, Boston, 1995.
J. E. Kasser, What Do You Mean, You Can't Tell Me How Much of My Project Has Been Completed?, INCOSE 7th International Symposium, Los Angeles, CA, 1997.
J. E. Kasser, A Framework for Requirements Engineering in a Digital Integrated Environment (FREDIE), Proceedings of the Systems Engineering, Test and Evaluation Conference, Brisbane, Australia, 2000.
J. E. Kasser, The Cataract Methodology for Systems and Software Acquisition, SETE 2002, Sydney Australia, 2002.
J. E. Kasser, Isn't the acquisition of a System of Systems just a simple multi-phased time-ordered parallel-processing process?, 11th International Symposium of the INCOSE, Las Vegas, NV., 2002.

J. E. Kasser, Object-Oriented Requirements Engineering and Management, The Systems Engineering Test and Evaluation (SETE) Conference, Canberra, 2003.
J. E. Kasser, Reducing the cost of doing work by an order of magnitude (by applying systems thinking to systems engineering), 21st Centre of Excellence Workshop: Challenges for life-based systems development, Tokyo, Japan, 2006.
J. E. Kasser, W. Scott, X.-L. Tran and S. Nesterov, A Proposed Research Programme for Determining a Metric for a Good Requirement, Conference on Systems Engineering Research, Los Angeles, CA., 2006.
J. E. Kasser, X.-L. Tran and S. Matisons, Prototype Educational Tools for Systems and Software (PETS) Engineering, Proceedings of the AAEE Conference, 2003.
J. E. Kasser and V. R. Williams, What Do You Mean You Can't Tell Me If My Project Is in Trouble?, First European Conference on Software Metrics (FESMA 98), Antwerp, Belgium, 1998.
G. Kotonya and I. Summerville, Requirements Engineering processes and techniques, John Wiley & Sons, Chichester, 2000.
P. Miller, Taming the Expanding System Complexity of a Commercial Product: Embracing Systems Engineering Principles in a Commercial Development Environment, First Regional Symposium of the Systems Engineering Society of Australia, INCOSE Region 6 (SE98), Canberra, Australia, 1998.
MIL-STD-499B, Mil-STD-499B Systems Engineering, United States Depeartent of Defense, 1992.
G. Myers, The Art of Software Testing, Wiley, 1979.
T. Peters, Thriving on Chaos: Handbook for a Management Revolution, Harper and Row, 1987.
T. Peters and N. Austin, A Passion for Excellence, Warner Books, 1985.
T. J. Peters and H. R. Waterman, In Search of EXCELLENCE, Harper and Row, 1982.
P. P. Reid, W. D. Compton, J. H. Grossman and G. Fanjiang, Building a Better Delivery System: A New Engineering/Health Care Partnership, The National Academies Press, 2005.
T. J. Rodgers, W. Taylor and R. Foreman, No-excuses Management, Doubleday, 1993.
W. W. Royce, Managing the Development of Large Software Systems, IEEE WESCON, 1970.
A. J. Shenhar and Z. Bonen, The New Taxonomy of Systems: Toward an Adaptive Systems Engineering Framework, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, (1997), Vol 27, No 2, 137 - 145.
X.-L. Tran and J. E. Kasser, Improving the recognition and correction of poorly written requirements, The Systems Engineering Test and Evaluation (SETE) Conference, Brisbane, Australia, 2005.
M. W. Wynne, Policy for Systems Engineering in DoD, Memo by Undersecretary of Defense for Acquisition, Technology, and Logistics, 2004.