A Framework for Requirements Engineering in a Digital Integrated Environment[1]

Joseph Kasser
Systems Engineering and Evaluation Centre
University of South Australia,
Mawson Lakes Campus, Room F-37
Mawson Lakes, SA, 5095
Australia
Email: Joseph.Kasser@unisa.edu.au

Abstract

The current systems and software development life cycle (SDLC) paradigm is that of a production system. The focus is on process and product. The Framework for Requirements Engineering in a Digital Integrated Environment (FREDIE) is a potential tool arising from the Anticipatory Testing concept which views the SDLC from the perspective of Information Systems, and the application of Knowledge Management and modern Quality theory. The FREDIE has the potential to implement significant cost reductions in the SDLC. This paper introduces the FREDIE concept, outlines how the cost reductions can be achieved by means of a tool that could become the basis for the next generation of software project management tools and presents some use cases of the FREDIE tool.

Introduction

The SDLC has evolved several methodologies since the early days of the Waterfall model. One of them, the Spiral model (Boehm 1988) placed 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?" (Kasser 1997).
- "What Do You Mean You Can't Tell Me if My Project is in Trouble?" (Kasser and Williams 1998).

Another flaw in the paradigm is that the SDLC is characterized by large cost overruns, schedule slips, and dramatic performance deficiencies in weapon, C4I, and automated information systems (DoD 1995). The reasons for these failures are varied (Kasser and Williams 1998) however major contributors to the failures are poor requirements and poor requirements engineering management. 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 requirement 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).

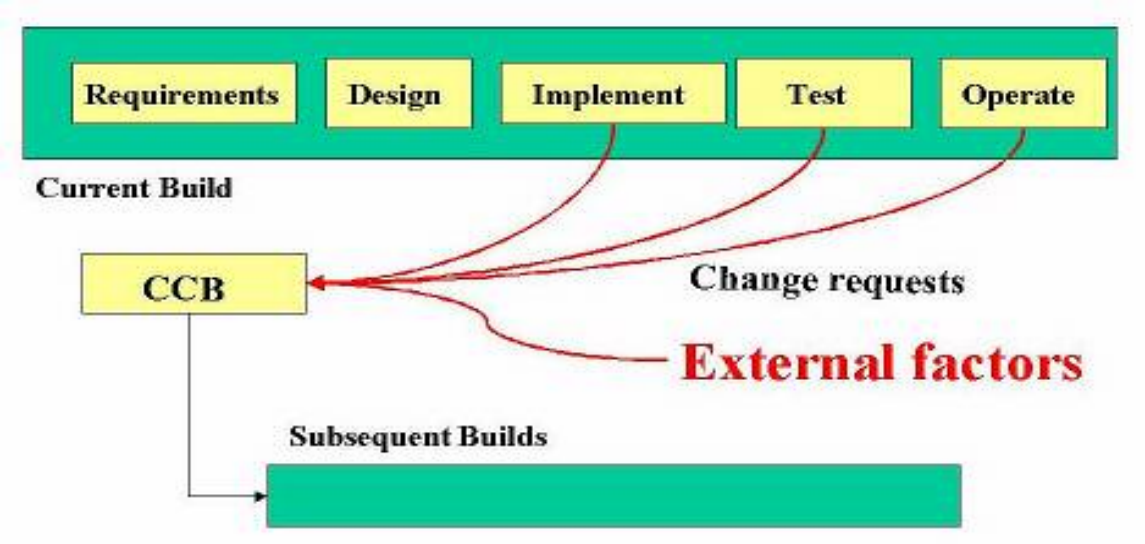
[1] This work was funded from the DSTO SEEC Centre of Expertise Contract.

Anticipatory Testing

Anticipatory Testing combines prevention with testing and is based on the recognition that prevention is planned anticipation (Crosby 1981). The Anticipatory Testing approach (Kasser 1995) 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. The Anticipatory Testing approach has the potential to provide better answers to the two questions posed above, facilitate change management and presents an opportunity for an improvement in program management at least as great as that obtained by the introduction of PERT[2].

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) and as "fitness for use" (Juran 1988). 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. Having to consider both the functionality and Quality criteria components of a requirement should tend to ensure that requirements are measurable and hence verifiable at the time of acceptance. Then later when the requirement is decomposed into subsystem requirements the flow down of measurable subsystem requirements should also tend to ensure that the capability provided by the system meets the performance required by the customer.

Figure 1 Anticipatory Testing view of the SDLC



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* and must not be considered

[2] Developed by the United States Department of Defense as a management tool for complex military projects. PERT is an acronym for "Program Evaluation and Review Technique".

independently. In addition, every one of the systems changes over time. From the Anticipatory Testing perspective **the 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).**

Figure 1 presents this concept by showing the traditional Waterfall methodology sequential elements connected via a CCB that allocates the implementation of requirements and subsequent changes to Builds in which:

- **Engineering** converts user needs into functionality and Quality criteria (requirements) and groups functionality into sets (Builds).
- **Management** ensures that Builds are implemented in a phased manner.

Change Management

From the Information flow perspective, the processes of accepting prospective requirements (before the baseline is set) and change requests (after the baseline is set) are identical and contain the following steps:

- Prioritize the requirement/change.
- Determine if a contradiction exists.
- Perform an impact assessment using an Integrated Product and Process Team (IPPT). The impact assessment must:
  - Estimate the cost/schedule to implement.
  - Determine the cost/schedule drivers - factors that are responsible for the greatest part of the cost/schedule.
  - Perform a sensitivity analysis on the cost/schedule drivers.
  - Determine if the cost drivers are really necessary and how much modification can be made by negotiating the requirement with customers based on the results of the sensitivity analysis.
- Make the decision to accept, accept with modifications, or reject.
- Notify the originator.
- Document the decision(s) in the requirement repository.
- If the requirement/change is accepted, allocate the implementation to a specific Build modifying the Work Breakdown Structure (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. In the existing production paradigm, this information tends to be contained in several different and usually unconnected tools: Requirements Management, Project Management, WBS, Configuration Control, and Cost Estimation, etc. This information, herein named Quality System Elements (QSE) includes but is not limited to:

- **Unique identification number** - the key to tracking.
- **Requirement -** the imperative statement containing both the required functionality and its corresponding Quality criteria or other form of representation.
- **Traceability to source(s) -** the previous level in the production sequence.
- **Traceability to implementation -** the next level in the production sequence. Thus requirements are linked to design elements, which are linked to code elements, and so on.
- **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.
- **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

requirement avoids accepting requirements that are either impossible to verify or too expensive to verify.
- **Risk -** any risk factors associated with the requirement.
- **Keywords -** allow for searches through the database when assessing the impact of changes.
- **Production parameters -** the Work Breakdown Structure (WBS) elements in the Builds in which the requirements are scheduled to be implemented.
- **Testing parameters -** the Test Plans and Procedures in which 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.

The FREDIE Paradigm

This paper proposes expanding the traditional RTM into a database represented by the set of Quality System Elements (QSE) to be stored in a Framework for Requirements Engineering in a Digital Integrated Environment (FREDIE) instead of the several separate tools currently in use. This database and its Agents[3] would be the next step in the evolution of Requirements Engineering; 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).

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 ambiguity in poorly written requirements is minimized because the rational for the requirement and the verification methodology are documented early in the process.
- Unrealistic and unverifiable requirements are not imposed on the developer by virtue of the testing parameters.

Access to the QSE via the FREDIE Agents will allow decisions to be made more rapidly and effectively. Thus the concept will improve the shared meaning as well as the three dimensions of Requirements Engineering success (El Emam and Madhavji 1996):

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

Use Case Scenarios

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

Change request - impact assessment

When a change request is received, the process outlined above is followed. Once the specific WBS element[4] and/or requirement (as appropriate) affected by the change is 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.

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

[3] Entities that operate on the contents on the database.

[4] Implementing the requirement.

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).
- 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.

Categorized Requirements in Process (CRIP)

The Categorized Requirements in Process (CRIP) approach (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. The CRIP approach follows the following procedure:

- **Develop categories for the requirements** - use factors such as complexity", "cost", and "priority" as appropriate.
- **Identify and set up the rules for 10 distinct ranges within each category** – the ranges represent a partitioning of the category into a number of ranges. A simple 1 to 10, or A - J is enough. The rules for determining the category must not change over the SDLC.
- **For each category, place each requirement into a range** - it is permissible to move a requirement from one range to another should the initial determination be found to be incorrect.
- As the SDLC progresses, for each category of requirements, record the state of each of the requirements in the FREDIE database – a requirement must reside in any one and only one of the following states at any time:
  - **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 a Build containing the implementation of the requirement.
- At each reporting milestone, monitor the *changes* in the *state* of each of the requirements *between* the SDLC reporting milestones - the typical project management view should be used, namely:-
  - **Expected** from last time
  - **Actual** achieved
  - **Planned** for next time

The summaries may be presented in graphical or Table format as shown in Figure 2. Each cell in the table contains three values (Expected, actual and planned). 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:

- The cell in the "Identified" column for category B shows that the project planned that 43 requirements would be identified, but only 2 were actually identified and the project expects to identify 4 in the next reporting phase. Something may be wrong here!
- In category B, it was expected that the part of the system implementing 12 requirements would go into test in the last reporting period, but only 5 made it into test and none are planned for the next reporting period. Again, something is wrong.

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. For example, 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 poor buyer requirements (requirements creep).

Figure 2 CRIP Chart for Category X

	Identified	In production	Com-pleted	In test	Accepted
A	26-23-4	6-5-9	5-12-45	12-5-1	10-0-20
B	43-2-4	34-6-9	5-12-3	12-5-0	0-0-0
C	12-5-46	2-9-18	4-4-4	0-0-0	0-0-0
D	5-12-5	4-9-12	2-2-2	0-0-0	0-0-0
E	6-12-9	12-9-4	5-9-15	0-0-0	0-0-0
F	5-12-34	6-12-4	8-12-8	0-0-0	0-0-0
G	5-12-5	1-1-1	6-9-12	0-0-0	0-0-0
H	6-9-19	2-9-10	6-9-15	0-0-0	0-0-0
I	12-5-3	6-9-12	5-4-3	0-0-0	0-0-0
J	0-0-0	0-0-0	0-0-0	0-0-0	0-0-0

Implementing the CRIP approach in a FREDIE could help build "prevention" into the tool. It would be straightforward to build intelligent agents to automate problem identification and assist human interpretation of the more subtle trends identified by FREDIE agents.

C4ISRAF

The Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) Architecture Framework (C4ISRAF 1997) is becoming increasingly prominent in Western defence circles. C4ISRAF is a response to unprecedented change in the military environment. It is intended to ensure that the architectures developed by geographic and functional unified Commands, military Services and defence Agencies are interrelatable between

and among the organizations' operational, systems, and technical architecture views, and are comparable and integratable across Joint and multi-national organizational boundaries. It is based on the assumption that to deal with the challenges of unprecedented change one must build around a problem invariant, and the "architecture" is the right invariant to use. The C4ISRAF design approach is to create building blocks from three architecture views (operational, technical and systems) each of which should be modular, reusable, and decomposable. The blocks are then to be assembled within the architecture using a structured top-down design and a piecemeal or "to-hand" implementation approach. In summary, the C4ISRAF paradigm:

- Is a product-oriented paradigm.
- Assumes requirements can be extracted from concept scenarios.
- Does not address 'poor requirements'.
- Focuses on a framework for design and implementation.
- Does not explicitly address Change control.

For a nation to gain the competitive edge through C4ISRAF, an implementation needs the ability to:

1. Design better building blocks against each of the three architecture views, cheaper and in a cost-effective manner.
2. Assemble and reassemble blocks into powerful and effective systems faster than a potential adversary.

FREDIE, by virtue of its ability to facilitate the handling of change, may help provide the framework for gaining that competitive edge in C4ISRAF.

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.

References

Boehm, B.W., "A Spiral Model of Software Development and Enhancement," *Computer*, May 1988, pp. 61-72.

Crosby, P.B., *Quality is Free*, McGraw-Hill, 1979.

Crosby, P.B., *The Art of Getting Your Own Sweet Way*, Second Edition, McGraw-Hill Book Company, 1981, p 131.

C4ISR Architecture Framework, Version 2.0, US Department of Defense, December 1997.

Department of Defense (DoD), *The Program Manager's Guide to Software Acquisition Best Practices*, Version 1.0, July 1995.

El Emam, K., Madhavji, N., "An Instrument for Measuring the Success of the Requirements Engineering Process in Information Systems Development", *Empirical Software Engineering*, 1, pp. 201-240, 1996.

El Emam, K, Hoeltje, D., "Qualitative Analysis of a Requirements Change Process", *Empirical Software Engineering*, 2, pp. 143-207, 1997

Glass,R.L., *Building Quality Software*, Prentice Hall, Englewood Cliffs, NJ., 1992.

Jarke, M., Meta Models for Requirements Engineering, *Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banf, Alberta, Canada, 1996, http://ksi.cpsc.ucalgary.ca/KAW/KAW96/jarke/Jarke.html, last accessed June 22, 2000.

Juran, J.M., *Juran on Planning for Quality*, The Free Press, 1988, p 11

Kasser, J.E., *Applying Total Quality Management to Systems Engineering*, Artech House, 1995.

Kasser, J.E., "What Do You Mean, You Can't Tell Me How Much of My Project Has Been Completed?", *The 7th Annual Symposium of the International Council on Systems Engineering (INCOSE)*, Los Angeles, CA, 1997.

Kasser, J.E., Williams, V.R., "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.

Kasser, J.E., "Using Organizational Engineering to Build Defect Free Systems, On Schedule and Within Budget", *PICMET*, 1999.

Author

Dr. Kasser is both a DSTO Associate Research Professor at the University of South Australia (UniSA) and a Distance Education Fellow in the University System of Maryland. He performs research into improving the acquisition process and teaches systems and software engineering subjects. Prior to taking up his position at UniSA, he was a Director of Information and Technical Studies at the Graduate School of Management and Technology (GSMT) at University of Maryland University College (UMUC). There, he was responsible for the Master of Software Engineering (MSWE) degree and the Software Development Management track of the Master of Science in Computer Systems Management (CSMN) degree. He is a recipient of NASA's Manned Space Flight Awareness Award for quality and technical excellence (Silver Snoopy), for performing and directing systems engineering. Dr. Kasser also teaches software engineering via distance education.