Reducing Project Re-Work By The Use Of Model Based Systems Engineering Processes and Tools

Steven Saunders
Raytheon Australia Pty Ltd
PO Box 165, North Ryde
NSW 2113, AUSTRALIA
Phone: +612-8870-6555
Fax: +612-8870-6599
ssaunders@raytheon.com.au

ABSTRACT

It is recognised that errors in requirements, injected into a project during the requirements definition phase, have the potential for significant project re-work later in the project. Project re-work may result in schedule and cost overruns affecting project profitability and useability. Sources of requirements defects include the inability of the human mind to comprehend highly complex systems and to convey this into understandable specifications, the inability to view all aspects of the system in a coordinated manner and poor specification writing practices to name a few. In this paper, the author proposes that another source of requirements defects is due to systems engineers starting their process with requirements analysis rather than functional analysis. The implementation of a Model Based Systems Engineering (MBSE) environment and methodology is described along with quantitative results based upon analysis of project requirements specifications developed over the past five years. As the current Model Based Systems Engineering environment was deployed two years ago, sufficient data exists to identify trends in specification quality both before and after deployment of MBSE. Current analysis presented indicates an emerging trend to real project savings for current projects using the deployed Model Based Systems Engineering methodology. Initial measurements indicate a 68 percent reduction in specification defects is being realised in current programs.

INTRODUCTION

Many studies have been conducted into the causes of project cost over runs and failures. A common cause identified results from incorrectly or poorly defined system requirements at the start of a project [Kasser and Williams 1998]. It is also understood that errors injected early in the development cycle have a higher potential cost impact to projects than defects injected later [Sampson 2000]. Techniques have been identified that enables quantification of requirement defects [Gilb 2002], however only an estimate of their impact on project costs can be made. Compounding the likelihood of defects in requirements specifications are limitations of human's short term memory, making it difficult to comprehend all aspects of complex systems currently being defined in industry. Research shows that humans find it difficult to hold sufficient information to work on more than around seven or so non-correlated pieces of information concurrently [Kline 1995].

Raytheon Australia suspected a problem with the systems requirements definition process (ie., it was allowing incomplete, inconsistent, ambiguous and un-verifiable

requirements to be generated). To address these problems, an improvement plan was initiated in 2001 in order to reduce the number of requirements defects being introduced early in the project life cycle. A three pronged strategy was implemented to;

a) Update processes to break down system complexity in order to assist the derivation of a balanced view of complex system solutions,
b) Deploy a modern systems engineering tool which is compatible with the new process, and
c) Deploy staff training on the process and tool.

The overall goal of the improvement plan was to re-align the thinking and support environment to the engineering of systems solutions rather than the generation of documents such as specifications. Key in achieving this goal was the implementation of a model based systems engineering methodology. More recently, it was decided to base this model upon the FRAT model [Mar 2002]. The FRAT model implemented enables any system to be viewed in four concurrent views; the Functional, Requirements, Architecture and Test views. Simplification of system complexity to a minimum of four views assists the systems engineer to comprehend the system to be designed.

At the end of 2002, a Raytheon Australia six sigma project was run to complete the improvement plan and to quantify the effects of any improvements. As a result of this activity, a 68% reduction in requirement defects is currently being realised. Whilst the real project cost savings of this reduction in defects can only be validated after current projects are completed, it is logical to expect a reduction in program re-work over the life of current and future programs.

ROOT CAUSES OF REQUIREMENTS DEFECTS

Using historical data from actual projects, the author has conducted an analysis of project re-work from past project defect reports. Where causes of defects were traced back to system requirements, the requirements were analysed to determine the root cause. Where system requirements contributed to the project defect, the requirement was considered defective. In the sample taken, a majority of requirement defects could be attributable to;

a) Un-verifiable requirements,
b) Ambiguous requirements,
c) Orphan requirements (not related to a system function),
d) Missing requirements, and
e) Functional behaviour not understood.

Many other sources of requirement defects exist [Gilb 2002]. However, for the purposes of addressing high impact root causes, the above factors were prioritised for correction.
Missing or incorrect requirements can also be attributed to poor requirements elicitation techniques, often driven by schedule pressure to get a specification delivered. Heydt provides a list of elicitation techniques which may be used during requirements definition [Heydt 2002].

DEFINING A REQUIREMENTS DEFINITION METHODOLOGY

Requirements Or Functional Analysis First?

Historically, many texts and literature have led to an interpretation that the systems engineering process begins with requirements analysis. This then is followed by functional analysis. Industry guidelines and standards did little to dispel this interpretation (for example IEEE 1220-1994). Experience on projects confirms the view that rather than defining requirements first followed by functions, systems engineering involves functional analysis followed (if not done in parallel) by requirements analysis. This is logical as functions define what the system must do whilst requirements define how well to do these functions. In effect, there can be no functional requirement without a corresponding function (and visa versa).

This premise has been tested on historical projects where specifications were found where requirements without corresponding functions have led to confusion over the meaning of the requirement. Whilst no quantitative data currently exists to show these issues resulted in project re-work, there are some indicators in the sample taken by the author indicating the misunderstanding of the real requirement is a direct relationship to the lack of a corresponding function in the system functional model. Further support to the concept that functional analysis leads requirements analysis has been identified [Mar 2002]. Mar indicates the misinterpretation of requirements analysis leading functional analysis is the result of systems engineering practitioners reading and following unclear guidelines verbatim.

Buede [2002] provides an excellent example of the power of functional analysis as the first step in the systems engineering process in his paper on the Concepts of Systems Engineering as Practiced by the Wright Brothers. The design process used by the Wright Brothers involved a “function-space-search” rather than a “design-space-search”. Buede claims the Wright Brothers were successful in developing the first powered flight by man where others had failed partially due to the fact they focused on functions before requirements and design.

Extending the Functional - Requirements View

Recognising functional requirements and functions cannot exist in isolation leads to the concept that there is a requirements and functional view of any system. However, there are more views required to fully define a system. A minimum of four views have been identified [Mar 2002].

a) The Functional view describes what the system does (behavioural model),
b) The Requirements view describes how well the system performs the functions,
c) The Architectural view (or answers) describes how the system is built,
d) The Test view describes how the system is to be tested (or verified).

Mar refers to this four view model as the FRAT model, as depicted in Figure 1. The order of the letters in the acronym FRAT are also important in that this defines the preferred order of systems engineering, Functional followed by Requirements, Architecture then Test views. The four views in the FRAT Model should be considered the minimum required to accurately define a solution to a system at any level of decomposition. However, additional views are often required to more accurately capture the need (eg. Operational View, Logistics View etc).

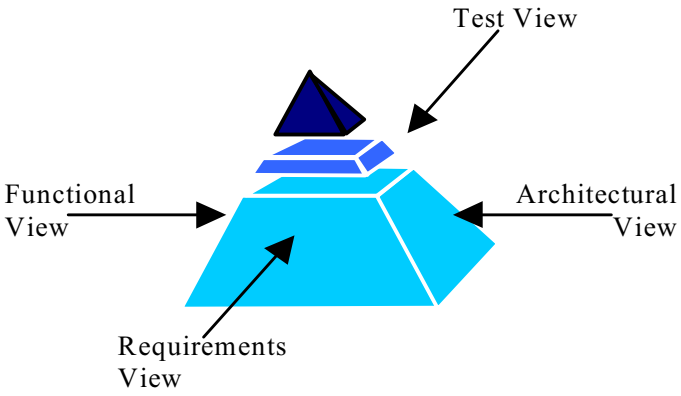

Figure 1 FRAT Model

Analysis of historical specifications shows instances where it was evident the systems engineer had overlooked the purpose of requirements definition and focused on writing requirement statements. In some instances, the decoupling of functions from requirements have led to ambiguous requirements and potential later project re-work.
For this reason, it was decided to focus future systems engineering on the use of a model based approach to systems engineering. This model based systems engineering approach would be consistent with the FRAT model.

SELECTION OF A SYSTEMS ENGINEERING TOOL

Requirements Centric Systems Engineering Tool
The term Requirements Centric Systems Engineering (RCSE) is used to describe Systems Engineering where the primary product is considered to be specifications and requirements traceability. RCSE is characterised by the use of tools such as wordprocessors or databases to capture requirements. Any behavioral modeling such as logical or functional modeling is performed in separate tools with no link to the Requirements Management tool.

RCSE does not naturally support the concept of understanding the functional behaviour of a system as the systems engineer must manually correlate data from different views potentially developed in different tools. In instances where less experienced engineers become involved, this can result in a specification document with well-formed requirement statements but with little consideration of the functional behaviour of the system to be realised. This specification provides a false sense of progress as it may appear to be complete, however the important information, the required behaviour of the system, may be incomplete, inaccurate or conflicting. This process is depicted in Figure 2

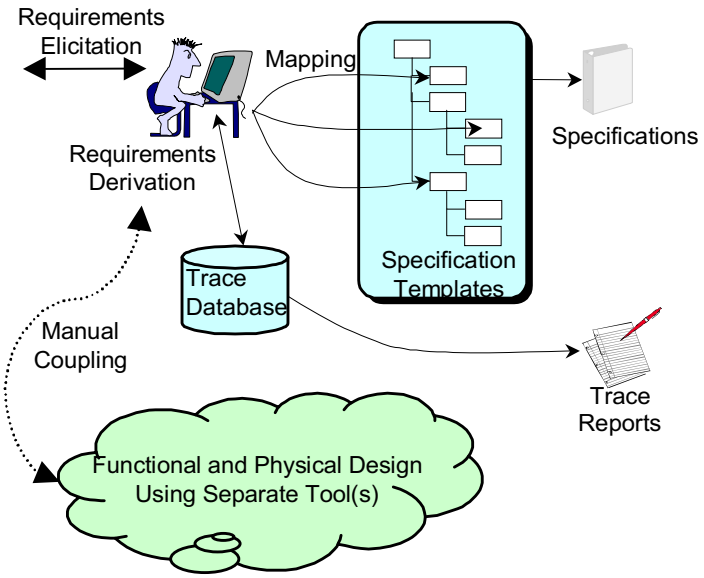

Figure 2 RCSE Environment

Model Based Systems Engineering

The term Model Based Systems Engineering (MBSE) is used to describe the conduct of systems engineering where the primary product is the system model. Here the term "model" refers to a consistent linked set of data, which represents as a minimum, the four key views of the system in the FRAT model. The mindset of the systems engineer is to complete the model. Specifications and traceability are considered to be incidental or artifacts of the process rather than the product of the process. This process is depicted in Figure 3.

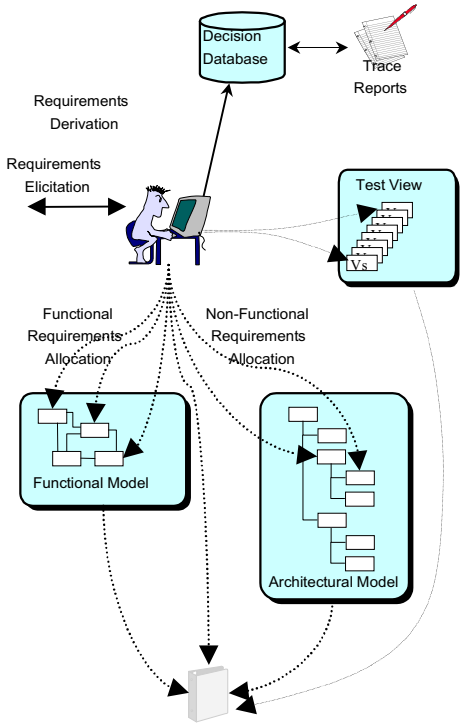

Figure 3 Model Based SE Environment

Use of a Model Based Systems Engineering Tool

Prior to selecting and deploying a systems engineering tool for use in the organisation, consideration was made to ensure the tool complements the MBSE processes. Effort has been applied to lessen the focus on the specification artifacts and to promote the function-space search approach to the systems engineer. Features have been built into the tool and environment to enable the artefacts of the engineering process (the documents and trace reports) to be generated from simple pull down menus.

The current systems engineering environment supports concurrent functional analysis, requirements analysis, synthesis and test (all four views in the FRAT model). Figure 4 provides a screen capture from the deployed systems engineering tool showing four windows, each providing a view into the FRAT model. In the functional view, the systems engineer may view hierarchical and/or functional flows to help characterise the functions and behaviour of the system. As the functional model matures, requirements are derived from customer requirements and traced to functions by simple drag and drop operations. As requirements are further analysed, changes/additions to the functional model are made. This iteration between functional and requirements views continues, building the functional model and the completeness of the requirements set. Experience from the systems engineers shows this is vastly more productive than sitting in front of a specification document template and trying to manufacture requirements.

As the functional model matures, the physical realisation can be constructed in the architecture view. Non-functional requirements such as power, weight, quality requirements are imposed on physical elements. In parallel with the definition of requirements, Raytheon Australia has dictated the capturing of verification statements for each requirement. Verification statements are objects in the systems engineering database which provide a clear statement on how each requirement is envisaged to be verified. Initially this was seen by systems engineers as an added burden however it has been found that this rigour has minimised the number of requirements that are specified in an un-verifiable way. By-products of this step are;

a) the initial test specification is drafted at the same time as the requirements baseline is being formulated, and
b) removal of a second iteration at interpreting the requirement and verification approach is avoided, saving duplication of effort.

Using the tool, relationships between views is clearly evident leading to;

a) a higher level of understanding on what the system does (behavioural model),
b) how requirements are related,
c) how requirements are to be verified,
d) whether the conceptual solution is realisable,
e) identification of missing functions, and
f) identification of missing requirements.

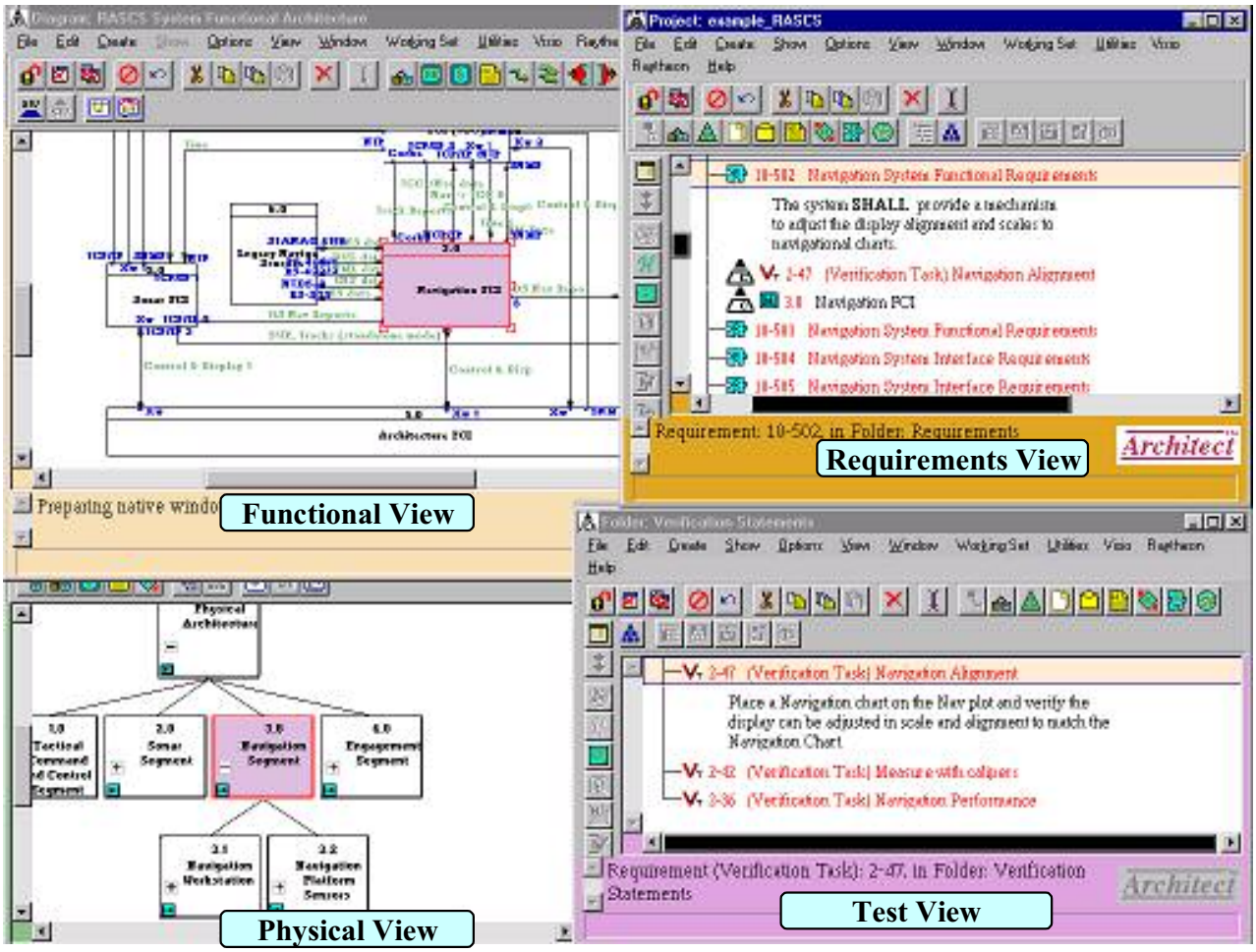

Figure 4 Screen Capture of the Systems Engineering Tool Showing all Four Views in the FRAT Model

Deployment and Training

Concurrently with the deployment of revised processes and tools, a training program has been deployed to further emphasise the "function-space" solution search concepts and to break down pre-conceived notions that systems engineers "just write" specifications. Emphasis is being placed on systems engineers doing value added engineering and having documents generated for them from the model.

Current uptake on the concepts has been very encouraging with engineers readily adapting to the model based approach. It is being found that a higher focus is being placed on understanding the behaviour of the systems currently in development which in itself is driving out many requirements and conflicts that may not have been evident if model based systems engineering was not being adopted.

MEASUREMENT OF RESULTS

A simplified model of sources of errors and the phases these errors are injected into the end product is provided in Figure 5.

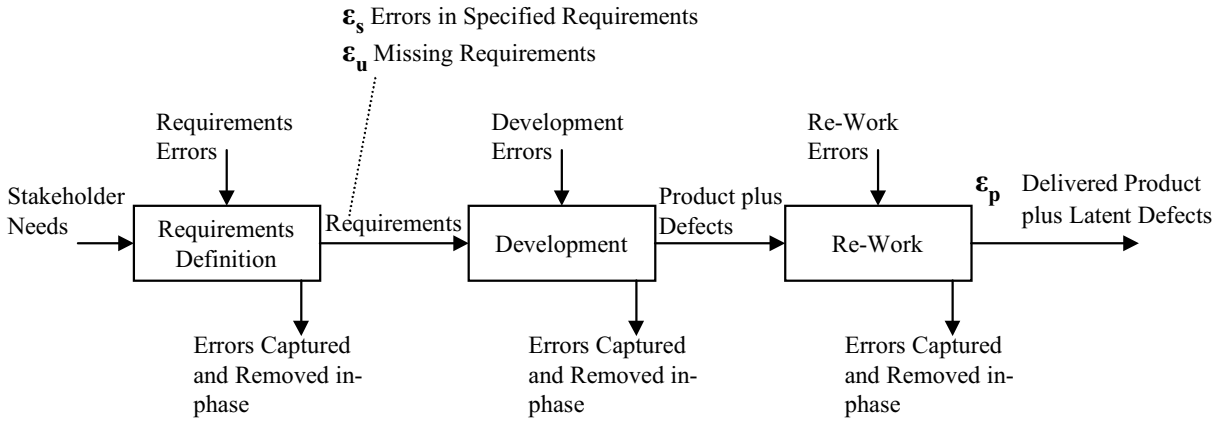

Figure 5 Model of Sources of Errors in an Engineering Development

In this model, stakeholder needs are transformed into system requirements in the requirements definition phase. During this phase, requirement defects are introduced and are manifested in the form of;

- $\epsilon_s =$ Errors in specified requirements (e.g. ambiguous, conflicting, un-verifiable requirements etc), and
- $\epsilon_u =$ Unspecified requirements errors (missing requirements).

During production, a percentage of requirement defects are promulgated into errors and re-work in the end product.

$$\text{Є}_p = k\ \epsilon_s + j\ \epsilon_u$$

Where Єp is rework cost and k and j are variables dependent upon the organisation, people and processes. As k and j are subject to change over longer periods of time, it is assumed k and j are constant over the period of the current process improvement activity.

In order to quantify the effect of process change, a set of specification quality rules were defined. These rules enable a count of non-compliances against each rule and provide a metric on specification quality. By definition, this process can only detect breaches of the rules for specified requirements (Єs). For the purpose of quantifying the savings to projects due to the requirements process improvement activity, only relative change was sought. It is assumed the defects related to unspecified deficits (Єu) remains constant. In addition, as only a relative change is being measured, the absolute value of the constant (k) is not required to be determined.

The rules applied to each requirement were;

a) All words are unambiguous to the weakest audience,

b) All words are clear with regard to the intent of the weakest of the intended audience,
c) The source of the requirement is provided and traceable,
d) The distinction between comments and requirements is explicit and obvious,
e) The requirement has an unique identifier,
f) The requirement is quantifiable,
g) The requirement is verifiable and
h) The requirement is free from design data.

Using a sample of past and present project specifications, the above rules were critically applied by independent reviewers and the results normalised to defects per requirement statement (note: a single requirement statement can have multiple defects if more than one rule is broken). This formed a baseline metric for requirements defects contained within specifications before the new processes and tools were deployed and current defect densities after deployment.

Results of the Defect Reduction Project

Specifications are only now starting to be created using the new processes and methodologies. Figure 6 shows the results of both the historical and current specification defect densities, normalised to defects per requirement statement (or "shall").

Despite the uncertainty introduced by a low sample size and interpretation issues between reviewers (especially as seen on the July 2000 Specification), there is a notable decrease in defect density after March 2002 when the new processes and environment were deployed. The average defect density before the improvement program was 1.05 defects/shall. After the improvement program, average defect density has reduced to 0.33 defects/shall, a 68 percent reduction in defects to specifications.

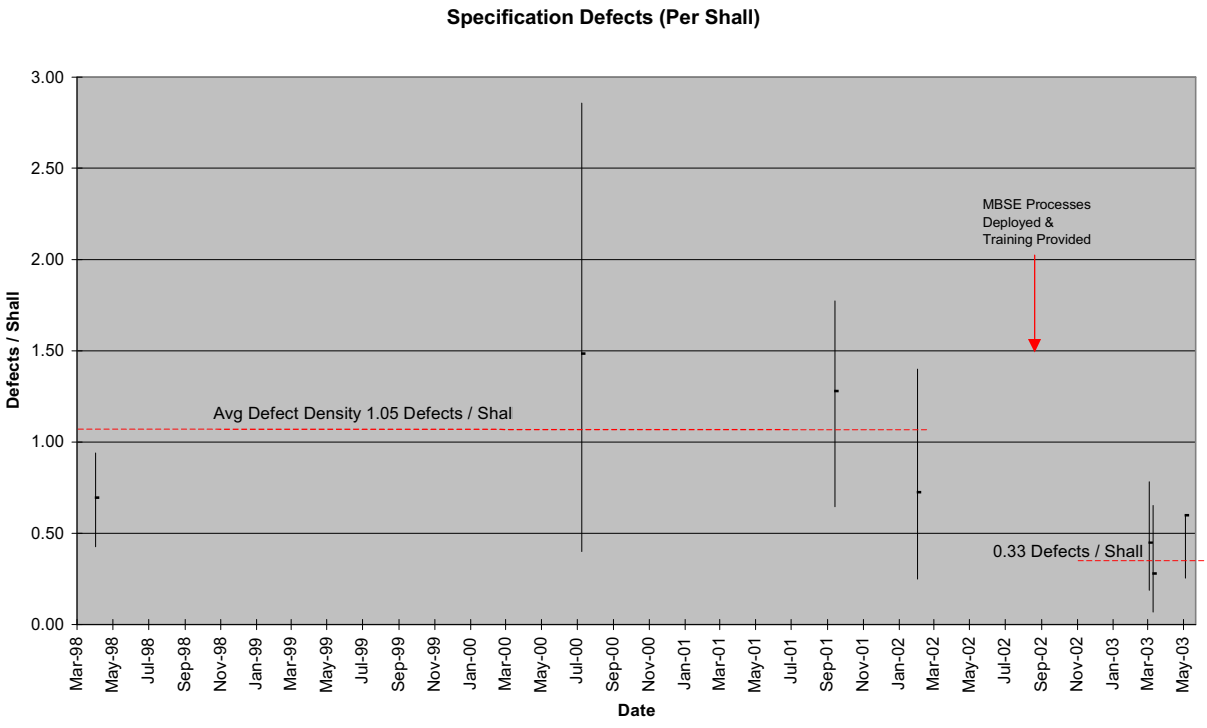

Figure 6 Measured Specification Defect Density Over a Range Of Specifications Since 1998

Future Work

Current work has focused on measuring the relative reduction in defects in specified requirement statements. The quantitative cost of requirements defects to projects has not been validated with a sufficiently large sample size to provide statistically reliable data. Further work is planned to analyse existing program data in order to quantify the ratio of specification defects to rework effort (in hours and hence dollars). When complete this activity will provide validated quantifiable cost of requirements defects to programs. This data will be used to identify further cost savings measures that may be implemented.

In addition, it has been found the use of a MBSE environment makes it easier to identify missing functions and requirements. It is expected this should lead to a reduction in missing requirements in future specifications. Further work is planned to identify and implement a plan to measure these savings (if any). When complete, a true indication of the overall return on investment will be made available.

CONCLUSION

Over the past decade, the author has seen many examples of requirements engineering where an undue emphasis has been applied to writing requirement statements in a specification and calling the end product the completion of requirements analysis. Instances can be found where the true understanding of what systems engineering is intended to do has been lost in the drive to baseline a specification document. The author proposes one cause of this has been the overlooked understanding that requirements definition actually involves understanding (modeling) the system and relationships between requirements, functional, architecture and test views of a system. Furthermore, it is suggested that functional analysis should precede requirements analysis. This is due to the need to define what the system must do before the performance of the system can be specified.

This paper has described the process of implementing a MBSE methodology and the deployment of a systems engineering tool which favours a model based systems engineering approach. In the new environment, the final specification and traceability are considered artifacts of the new process rather than the primary product. MBSE has been found to help systems engineers understand the behaviour and purpose of complex systems by providing correlated views into the system solution. It has been found a minimum of four views are required to adequately visualise any system.

Measurements of defects contained within specified requirements indicates a reduction of 68 percent has been achieved in specification defects over a two year process improvement program. Further un-measured improvements in specification quality are anticipated due to significantly improved visibility and understanding of complex system behaviour provided from the MBSE environment.

REFERENCES

Buede, Dennis, *The Concepts of Systems Engineering as Practiced by the Wright Brothers,* Proceedings of the INCOSE 2002 Symposium.

Gilb, Tom, *Requirements-Driven Management,* Draft book manuscript found at http://www.result-planning.com, 5 Sept 2002.

Heydt, Harold J., *Tools for Requirements Discovery, Creation, and Elicitation,* Proceedings of the INCOSE 2002 Symposium.

IEEE, *IEEE Trial-Use Standard for Application and Management of the Systems Engineering Process,* IEEE Std 1220-1994, 1994.

Kasser, Joseph, and Williams, Victoria R., *What Do You Mean You Can't Tell Me if My Project is in Trouble?*, Proceedings of FESMA98 Symposium, 1998.

Kline, Stephen J., *Conceptual Foundations for Multi-Disciplinary Thinking*, Stanford University Press, 1995.

Mar, Brian W. and Morais, Bernard G., *FRAT - A Basic Framework for Systems Engineering,* Proceedings of the INCOSE 2002 Symposium.

Sampson, Mark E., *Guiding Principles for Next Generation Computer-Aided Systems Engineering Tools,* Proceedings of the INCOSE 2000 Symposium.

ABOUT THE AUTHOR

Steve Saunders received his Bachelor of Electrical Engineering, from the University of Technology Sydney (UTS) with first class Honors in 1990. He has worked with Rockwell International, Boeing Australia and now Raytheon Australia on Australian Defence projects in various Systems Engineering, Design and Test roles.

Steve currently is the Systems Engineering Manager for Raytheon Australia Naval Systems Division and has a strong interest in improving System Engineering maturity and the agility of Systems Engineering to support the rapidly evolving technology environment within the Defence Industry.