Using a Rapid Incremental Solution Construction Approach to Maximise the Completeness and Correctness of a Set of Requirements for a System

Joseph E. Kasser DSc CEng CM MIEE, Stephen C. Cook PhD FIEE FIEAust
Systems Engineering and Evaluation Centre, F2-37
University of South Australia
School of Electrical and Information Engineering
Mawson Lakes Campus, South Australia 5095
Telephone: +61 (08) 8302 3941, Fax: +61 (08) 8302 4723
Emails: Joseph.Kasser@unisa.edu.au, Stephen.Cook@unisa.edu.au

Abstract. While the problem of poor requirements engineering and management has been documented since the dawn of systems engineering the continual documentation and discussion of the problem requirements has not resulted in a practical solution to the problem. This paper discusses applying the Blackboard methodology used in Artificial Intelligence to the development of a suite of Computer Enhanced Systems Engineering (CESE) tools that may be able to alleviate much of the problem. The paper also describes conceptual prototypes of the tools that have already been developed, their effect on alleviating the problem of poor requirements, and some concepts for future tools in the suite.

INTRODUCTION

The systems and software development industry is characterized by a paradigm of project failure (Standish 1995). 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?"* While the problem of poor requirements engineering and management has been repeatedly and widely discussed and documented for at least 10 years as a contributing cause of project failures (Hooks 1993; Kasser and Schermerhorn 1994; Standish 1995, Jacobs 1999; Carson 2001; etc.), the continual documentation and discussion of the problem of poor requirements engineering and management has not resulted in a practical solution to the problem.

At the same time as systems engineering is suffering the problem of poor requirements engineering and management, Kemp et al. (2001) write that the knowledge management community needs to address several essential issues immediately, 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.

This paper describes an evolutionary process for the continued evolution of knowledge management capabilities and also proposes to tackle the problem of poor requirements engineering and management using an approach that is based on the

- Blackboard methodology pioneered by Nii (1986),
- Agent based approach of using a suite of software products for rapid software evolution based on a domain model was presented as a way to develop software that is quicker and less expensive to maintain (Glover and Bennett 1996).

THE RISC APPROACH

The approach is named Rapid Incremental Solution Construction (RISC)[1]. It is

- an incremental multi-disciplinary approach based on the Cataract Methodology (Kasser 2003).
- 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 entire problem (or at least a major part of it) is eventually solved.

[1] The authors are very conscious of the risks involved in implementing partial solutions to a problem; hence the choice of acronym.

Figure 1a Start of Evolutionary Process for System Automation

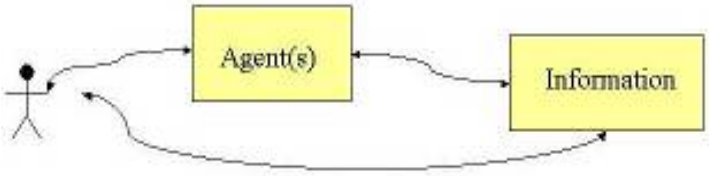

Figure 1b Software Agents Perform Some Automated Functions, the User the Others.

Figure 1. Evolutionary System Development

- An approach to solving the problem of poor requirements by developing a collaborative system that includes encapsulated tools, newly designed custom software, and human collaborators (Lander 1997).

The RISC approach is to provide concept demonstrators of the functionality needed in the form of a suite of simple next generation prototype Computer Enhanced Systems Engineering (CESE) tools or software agents having similar user interfaces and each performing a small range of functions. Some of these tools are known as Prototype Educational Tools for Systems and Software (PETS) engineering

Systems Engineering is focused on dealing with well-structured problems. However, the problem of poor requirements is complex and ill-structured problem, and hence not solvable by the traditional systems engineering process. One aspect of dealing with complex and ill-structured problems such as the problem of poor requirements is that an accumulation of knowledge in the system can often clarify the problem (Nii 1986). In our experience it also tends to suggest solutions. Nii suggests:

> *"Consequently a knowledge engineer needs to engage in exploratory programming, Exploratory programming, as defined by Beau Sheil (1983) is a 'conscious intertwining of system design and implementation.' Waterman (1985) notes that expert system building is accomplished in developmental stages ranging from research prototype to filed prototype until a fielded systems is evolved. That is, expert systems are also developed incrementally."*

This is the approach we have taken in developing the RISC suite as well as the individual tools. The evolutionary process is shown in Figure 1. The process begins with users interacting with information as shown in Figure 1a. The knowledge is in the users. Once the situational Use Case is monitored and understood, a software agent can be created to automate activities that are performed repeatedly as shown in Figure 1b. The evolution of the system can then be thought of as the migration of knowledge from the users to the agents.

The RISC Suite of CESE tools consists of a suite of tools The following CESE tools have already been developed

- The First Requirements Elucidation Demonstration (FRED) tool.
- The Communications Requirements Evaluation & Assessment Prototype (CREAP).
- The Operations Concept Harbinger (OCH).

FRED is designed to prevent the production of poorly written individual requirements and elucidate existing requirements. Kasser (2002) proposed a partial solution to the problem of poorly written requirements by describing a simple prototype of a software tool that could improve the wording of requirements. When released in October 2002, this tool was named FRED.

FRED performs a syntax check on the text of the requirement pointing out potential defects. In much the same way that a word processor grammar checker is used, it is up to the user to determine if the defect is there and how best to correct it. As more is learnt about defects in requirements wording, FRED can be upgraded to incorporate that knowledge. Future iterations of FRED might evolve into a product having similar features to the commercially available StyleWriter, the program that can analyse document and highlight all of its faults (at least according to the web site (StyleWriter 2002).

FRED can be used in two roles:

- To provide immediate feedback to requirements writers that the document that they are compiling has poorly worded requirements. In this mode it helps prevent the publication of poorly written requirements.
- To ensure that all existing requirements can be adequately tested, by identifying the individual requirements in a multiple-requirement paragraph of a Requirements Document. In this mode it facilitates the generation of a 'requirements to test' traceability matrix.

CREAP is more ambitious. It is targeted for use in a specify-to-inventory situation. It assists the writers of requirements by directing them to only specify requirements that are feasible in a given concept of operations. Kasser and Cook (2002) described this tool, which employs a frame-based approach (Cook et al. 2001) to reason about the feasibility of the requirements. CREAP is a frame-based requirements engineering tool (FBRET) that integrates military communications domain expertise and requirements engineering practice. Specifically, it ensures that the set of equipment selected to meet the requirements imposed on a communications systems for a rapid force deployment will constitute a viable military information system.

The OCH is the result of the application of information management technology to solving the problem of poorly written and articulated requirements as well as the effect of changing requirements for both single systems and Systems of Systems (Kasser et al., 2002). The OCH:

- is a tool for capturing user needs without the explicit use of "text-mode requirements";
- may be thought of as a multimedia Operations Concept Document that also contains measures of effectiveness for each operational scenario;
- helps to bridge the gap between the soft systems methodologies used in the early phases of the system development life cycle and the hard systems methodologies used in the construction of a system.

A MULTI-DISCIPLINARY APPROACH

The RISC approach to maximising the correctness and completeness of requirements uses techniques from several disciplines. These are:

- The concept that a document can be considered as a report from a database.
- Rapid prototyping of software.
- The object-oriented concept of inheritance.
- Expert systems and artificial intelligence.
- Effective Configuration Management of software products and processes.
- An adaptation of the Blackboard methodology used in Artificial intelligence.
- The suite of agents approach.
- The hierarchical perspective.
- The need for simple tools.

The concept that a document can be considered as a report from a database. For example, a Requirements Document can be considered as a printed view of a requirements database.

Rapid prototyping of software. Software engineering with the ability to quickly create software has developed a methodology known as rapid prototyping. Rapid prototyping can be used to allow customers to see and use the user interface of a complex software program, or to develop and incrementally release small software packages such as FRED. Copies of programs like FRED can be distributed among the practitioner community and the resulting feedback can be examined and used to construct an upgraded version.

The object-oriented concept of inheritance. Kasser (2000) stated, *"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 the requirements for the specific type of product. For example, a low earth orbiting (LEO) spacecraft will inherit a set of generic requirements that have been refined from the experience gained in building these 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."*

Expert systems and artificial intelligence. Cook (1990, 1991, 1993) discusses a structured document generation and processing shell, which represents the earliest reported knowledge-based system for computer-aided generation of equipment specifications. On completion of the specification elicitation, checking, and

editing functions, the tool was able to process the model of the specification held in the frame structure in Prolog to produce a written document in MIL-STD-490A format. Kasser (1992) introduced ELMER, a simple expert system based on a state machine that parsed text and had different ways of reacting to a word match. Wu and Lee (1993) discuss the construction of an object-oriented expert system for designing local area networks based on functional requirements. Sutcliffe et al (1998) describe a system that reuses requirements knowledge through decision models and generic models.

Effective Configuration Management of software products and processes is critical in conventional development. In rapid prototyping and RISC situations, configuration control is even more critical. Kasser (2002a) shows how configuration control can be used on both the product and process.

An adaptation of the Blackboard methodology used in Artificial Intelligence. Nii (1986) states that the blackboard model is an approach not a tool. The term blackboard is derived from the architecture of the system in which the processing activity is organised to resemble a group of experts reading the contents of, and then writing responses on, a blackboard. The basic approach to problem solving in the blackboard framework is to divide the problem into loosely coupled subtasks. Geymayr and Ebecken (1995) extend the definition of a blackboard system to one that consists of several knowledge systems that communicate through a blackboard and are controlled by an inference mechanism.

The suite of agents concept. The Agent based approach of using a suite of software products for rapid software evolution based on a domain model was presented as a way to develop software that is quicker and less expensive to maintain (Glover and Bennett 1996). Kasser (2000a) also described the concept of a suite of tools accessing a common expanded requirements database as an approach for tackling the problem of project failures due to poor requirements engineering and management.

The hierarchical perspective. The problem of poor requirements can be viewed from a hierarchical perspective. Kasser and Schermerhorn's (1994) requirements for writing requirements can be grouped into those that deal with

- single requirements, and
- sets of requirements.

Thus the application of blackboard-based expert systems and via a suite of software agents to the problem of poor requirements is motivated by the same reasons as Weiss and Stetter (1992) had for using the Hierarchical Organised Parallel Expert System (HOPES) that provided a framework for working with real-time parallel expert systems, namely

1. A single blackboard may become a bottleneck in the process of solving the global problem.
2. The use of the multiple-blackboard architecture reduces the complexity of the problem into a number of simpler problems.
3. Managing the multiple-blackboard-based projects is easier than managing a big complex project because the management is spread over several smaller projects if the appropriate methodologies are used. In this case the Cataract Methodology (Kasser 2002a) manages the development of the PETS suite of CESE tools and the object-oriented concept of abstraction is used to focus on the specific defect(s) addressed by a tool.
4. The limited functionality of each agent or tool provides a way to develop software that is quicker to develop and modify and less expensive to maintain (Glover and Bennett 1996).

In the RISC approach, the loosely coupled suite of blackboards becomes a suite of loosely coupled CESE tools as shown in Figure 2. Each tool operates on one or more elements in the hierarchy of defects in requirements. The users are the agents acting on the tool, and the tools are the agents operating on the central blackboard. The tools

- perform useful functions in themselves, and

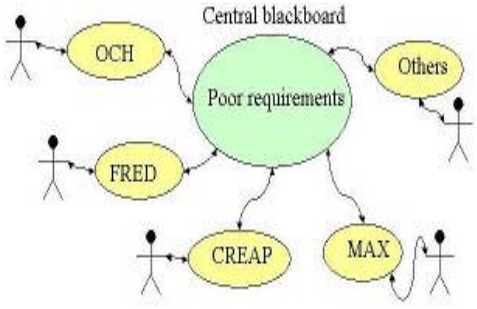

Figure 2 The RISC Approach to Multiple Blackboards

- serve as a discussion points about the scope and nature of the problem, and
- serve as examples of functionality for tool vendors to implement.

The knowledge application strategy is both top-down and bottom-up. The top-down strategy identifies the broad functionality of each of the specific tools, e.g. FRED elucidates poorly written requirements. The bottom-up strategy is driven by the results of applying the tool in the workplace. The first version of a tool uses knowledge provided by the constructors. As the tool is used, the knowledge transfers from the users to the developers in the form of positive comments in suggestions for upgrades and negative comments in the form of (constructive) criticism. This information finds its way to the central blackboard; in this case the PETS Suite Configuration Control Board (CCB). The CCB analyses the information and authorises changes in one or more of the tools as appropriate. For example, after some use of FRED, it was noticed that if there were a way to tell FRED to ignore a specific word in the next requirement, FRED's functionality would be improved. This situation arose because of inexactness in today's requirements. For example, consider the use of the word "and" in a requirement. A requirement such as "`DADS shall display the combined number of apples` **`and`** `oranges`" is a requirement to display a single total, hence the use of the word "and" is appropriate. On the other hand, a requirement such as "`DADS shall display the number of apples` **`and`** `oranges`" is two requirements, the first to display the number of apples, and the second to display the number of oranges. This is a defect in the requirement that complicates the building of traceability matrices and causes difficulties in the test and evaluation processes. In this instance the change is limited to a single tool. , The tool thus evolves under CCB control, becoming increasingly useful and over time, solves more and more parts of the problem. The suite approach allows for specific tools to be built or modified for testing hypotheses for improvements. Each hypothesis could be generated from a bottom-up or top-down source.

Issues with the use of words. The military standards in the form of MIL-STD 490B and its replacement MIL-STD-961D (1995) now used as a 'standard practice' states that the word "shall" defines a requirement. However; there seem to be other opinions on this subject in the United States of America (USA). The Federal Aviation Agency (FAA) has a plain language initiative and published guidelines for writing documents in plain language. The document (FAA 2000) states

> *"Shall" is one of those officious and obsolete words that has encumbered regulations and other documents for many years. The message that "shall" sends to the reader is, "this is boring material." "Shall" is imprecise. It can indicate either an obligation or a prediction. Dropping "shall" is a major step in making your regulation more reader-friendly many agencies already use the word "must" to convey obligations with no adverse legal effects.*
>
> *You can avoid "shall" by substituting "must" to indicate an obligation or "will" to indicate that an action will occur in the future. Be careful to consider which meaning you intend to communicate to your readers.*

However, the word "shall" is precisely defined in the Random House dictionary and thus is not ambiguous at all. Moreover, "must" does not convey who is responsible for making the system comply as clearly as "shall".

The actual word that is used is immaterial, as long as everyone agrees to use the same word for a specific meaning. Until there is an agreement on vocabulary, requirements engineering cannot move off the baseline. Tools like FRED may help to clean up these situations by strongly suggesting the use of specific words.

The blackboard system design process (McManus 1992) is shown in Figure 3. It is very similar to the rapid prototyping concept. The RISC approach based on using Concurrent Blackboard Systems is shown in Figure 4. It is very similar to, and based on, the Cataract methodology of Build planning (Kasser 2002b) for a single product and the RISC approach uses this methodology for managing the development of a suite of loosely coupled products that may be combined into a single multi-functional product sometime in the future. This is methodology is also the proven hardware approach of "build a little, test a little".

Integration is of the functionality and not necessarily of the tools. The software architecture is based on the arrangement and methodology of Kasser (1997) to facilitate the modularity.

The need for simple tools. Once the usefulness of the functionality of these tools is proven, the subsequent generation of tools may or may not combine functionality. There is much to be said for avoiding the bloatware of current office automation products. Moreover, the current requirements engineering tools have a

long learning curve, which sets a high threshold to be overcome when arguing for their adoption. Tools need to be simple to be used widely. The goal for the PETS suite of conceptual CESE tools is that they should be no more difficult to use than a slide rule and that each tool must help to improve the current situation.

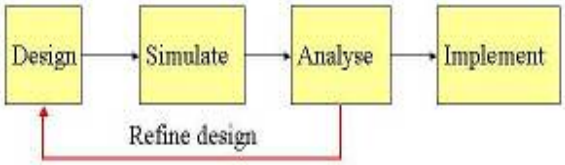

Figure 3 The Blackboard System Design Process

INTRODUCING MAX

This paper now proposes another tool in the PETS suite of CESE tools. This is a tool to maximize the completeness of a set of requirements, hence the name MAX.

The activities in a reference model of the process of producing a requirements document as derived from Cook (1993) may be stated as

1. Start by identifying and articulating the need;
2. Review previous requirements documents of similar systems for format and content;
3. Decide on the format of the requirements document;
4. Create the template for the document;
5. Write the requirements;
6. Test the quality of the written requirements;
7. End by releasing the first version of the document via the configuration control system.

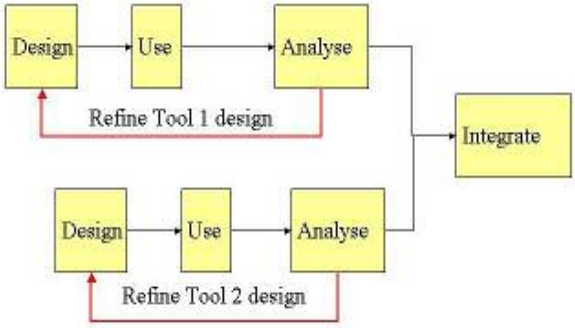

Figure 4 The RISC Design Process

The OCH can improve the articulation of the need (step 1). FRED can improve the wording of requirements and mitigate the effect of poorly written requirements (steps 5 and 6). MAX addresses steps 2, 3 and 4.

The format and content of documents for similar equipment is similar. Savvy systems engineers have used this knowledge to produce documents for new equipment by copying the format and marking up the documents for earlier equipment. While this simple process results in a more complete set of requirements than an engineer creating a document from nothing, it does not produce requirements that were missing from the earlier document. In general, these missing requirements tend to include the non-functional, environmental, transportation, and logistics, requirements. Thus the goals of MAX are to demonstrate that

1. the same functionality as the savvy engineer uses to produce a requirements document can be incorporated into a tool, and
2. the object-oriented concept of inheritance can be used to identify many, if not most, of the missing requirements.

MAX is broadly based on the knowledge gained in the construction and use of Specriter 3 (Cook 1993), a tool that was able to produce specifications for measuring instruments. MAX will also contain knowledge of requirement document formats and use them to produce requirements documents in the form of reports from the contents of the requirements database.

MAX will also have access to generic databases of non-functional requirements, will recognise which are appropriate and present them for tailoring into the requirements database of the system under consideration. The systems engineer will then have to view each requirement suggested by MAX and use it as is, delete it as inappropriate,

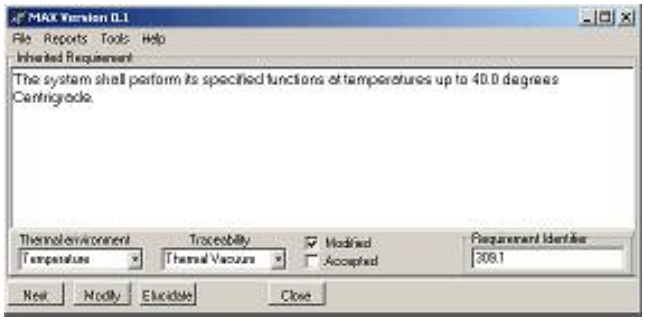

Figure 5 Typical Display From an early MAX Prototype

or mark it up via an interface such as the one shown in Figure 5. MAX will prompt the engineer if this task is not done. MAX will also take care of the traceability of these additional requirements.

MAX will be developed in a manner similar to FRED. In fact FRED or its functionality will be used to elucidate the wording of human originated requirements in MAX. Thus MAX and FRED can also be thought of as two agents operating on the same requirements database.

Adding non-functional requirements. Non-functional requirements are those that are necessary for the performance of a system but do not specify the functions performed by the system. For example environmental requirements specify the environment in which the system is to operate, not what it is to do in that environment. The hierarchy of environments for equipment is small. At the top level it consists of space, air, land, and marine. The marine environment can first be divided into above and below the surface and then into further subdivisions. Similarly, the land environment can be divided into indoor and outdoors, urban, jungle, desert, etc. This hierarchical arrangement can be viewed as an object-oriented hierarchical class structure. So if MAX knows what environment the system is to operate in, MAX can present the appropriate environmental requirements for consideration. MAX will also need some other smarts, such as knowing that the system will be transported for installation, and will know which transportation requirements to present for consideration.

Initial application. MAX's first application will probably be in the production of documents for communications systems. However, as in Specriter, the architecture will be generic and will be readily adaptable to other systems.

FUTURE CESE TOOLS

Other PETS are both under consideration and development for both the functional and object-oriented paradigms and perhaps combining them. The work of Wu and Lee (1993), which discussed using an object-oriented expert system for designing local area networks based on functional requirements, is under consideration. However there are important non-functional requirements that also need to be met in the installation of a communications system. One of these non-functional requirements is survivability in the event of catastrophic failures due to internal and external stimuli. The development of the PETS, MAX and the extension of the CREAP should provide insight into building an objected oriented tool for the design of communications systems based on a completer set of requirements than is generally available in today's system development paradigm.

SUMMARY

The RISC approach to solving the problem of poor requirements is the time-phased parallel process of building a suite of tools for performing specific tasks. This is similar to the standard systems engineering process of partitioning the desired functionality of the system into subsystems and then constructing and integrating each subsystem into the desired system. Once each tool is in use, new applications and modification are suggested. The suite then evolves until much if not the entire problem is solved.

Systems engineering currently does not begin to partition the functionality of the design until all requirements for the system are known. Thus systems engineering aims for a solution (the concept of the system that will solve the problem) that will be implemented via a reductionist approach in a time-ordered multi-phased parallel process. The RISC approach on the other hand partitions the problem into several sub-problems and begins implementing the sub-solutions in a time-ordered multi-phased sequential manner before the entire solution is known, evolving the sub-solutions into a complete solution.

While the RISC approach may not be applicable to solving all complex problems, in this application the results have been promising. FRED, CREAP, and the OCH provide capability that shows that parts of the problem of poor requirements can be alleviated by applying the appropriate technology.

AVAILABILITY OF THE TOOLS

The tools that have been tested and released, can be obtained by accessing the Systems Engineering and Evaluation Centre website (http://www.seec.unisa.edu.au) and following the link to Software Tools. The tools are provided free for educational purposes.

REFERENCES

Cook S.C. (1990), "Knowledge-Based Generation of Measuring Instrument Specifications", in *IMEKO International Symposium on Knowledge-Based Measurement*, 1st, Karlsruhe, FDR, pp145-152, 1990.

Cook S.C. (1991), *A Knowledge-Based System for Computer-Aided Generation of Measuring Instrument Specifications*, PhD thesis, City University, London, UK.

Cook S.C. (1993), "A Knowledge-Based Specification Generation System for Computer-Aided Production of Measuring Instrument Specifications", *Measurement*, Vol 11, pp 235-255.

Cook S. C., Kasser J.E., Asenstorfer J., "A Frame-Based Approach to Requirements Engineering", 11th *International Symposium of the INCOSE*, Melbourne, Australia, 2001.

FAA 2000, "Writing User-Friendly Documents - A Handbook for FAA Drafters February 2000", available at http://www.faa.gov/language/, last accessed October 14, 2002.

Geymayr J.A.B., Ebecken N.F.V. Fault-Tree Analysis: A Knowledge-Engineering Approach, IEEE Transactions on Reliability, Vol. 44, No 1., March 1995.

Glover, S.J., Bennett, K.H. "An Agent-Based Approach to Rapid Software Evolution Based on a Domain Model", *Proceedings of the International Conference on Software Maintenance*, page(s): 228 –237, 1996.

Hooks I., "Writing Good Requirements", *Proceedings of the 3rd NCOSE International Symposium, 1993,* available at http://www.incose.org/rwg/writing.html, last accessed November 1, 2001.

Jacobs S., "Introducing Measurable Quality Requirements: A Case Study", *IEEE International Symposium on Requirements Engineering*, Limerick, Ireland, 1999.

Kasser J.E., "ELMER: An Expert System Based on a Finite State Machine", *AMSAT Symposium*, Washington DC., 1992.

Kasser J.E., Schermerhorn R., "Determining Metrics for Systems Engineering", *Proceedings of the 4th NCOSE International Symposium*, San Jose, CA., 1994.

Kasser, J.E., "Yes Virginia, You Can Build a Defect Free System, On Schedule and Within Budget", *Proceedings of the 7th INCOSE International Symposium*, Los Angeles, CA, 1997.

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", *The Systems Engineering, Test and Evaluation Conference 2000 (SETE 2000)*, Brisbane, Australia, 2000.

Kasser, J.E. (2000a). "A Framework for Requirements Engineering in a Digital Integrated Environment (FREDIE)", *Proceedings of the Systems Engineering, Test and Evaluation Conference*, Brisbane, Australia, 2000.

Kasser J.E., "A Prototype Tool for Improving the Wording of Requirements", *Proceedings of the 12th International Symposium of the INCOSE,* Las Vegas, NV, 2002.

Kasser J.E, "Configuration Management: The silver bullet for cost and schedule control", *IEEE International Engineering Management Conference (IEMC 2002),* Cambridge, UK, 2002a.

Kasser J.E., "The Cataract Methodology for Systems and Software Acquisition, *the Systems Engineering Test and Evaluation Conference (SETE 2002),* Sydney Australia, October 2002b).

Kasser J.E. and Cook S.C. "The Communications Requirements Evaluation & Assessment Prototype (CREAP)", *Proceedings of the 12th INCOSE,* Las Vegas, NV, 2002.

Kasser J.E., Cook S.C., Scott W, Clothier J., Chen P., "Introducing a Next Generation Computer Enhanced Systems Engineering Tool: The Operations Concept Harbinger", *SETE 2002,* Sydney Australia, 2002.

Kemp L.L., Nidiffer K.E., Rose L.C., Small R., Stankowsky M., "Knowledge Management: Insights from the Trenches", IEEE Software, November/December 2001, pp. 66-68.

Lander S.E., *Issues in Multiagent Design Systems*, IEEE Expert, Volume: 12 Issue: 2, March-April 1997.

McManus J.W., *Design and Analysis Techniques for Concurrent Blackboard Systems*, Williamsburg, VA., Ph.D. Thesis; The College of William and Mary in Virginia, Sponsored by NASA. Langley Research Center, April 1992, Available at http://techreports.larc.nasa.gov/ltrs/PDF/phd-92-mcmanus.pdf, last accessed October 10, 2002.

MIL-STD-961D, Department of Defense Standard Practice for Defense Specifications, 22 Mar 1995, available at http://www.dsp.dla.mil/documents/961d.pdf last accessed October 15, 2002.

Nii H.P., "Blackboard Systems", Knowledge Systems Laboratory Report No. KSL 86-18, Knowledge Systems Laboratory, Department of Medical and Computer Science, Stanford University, 1986.

Sheil B., "Power Tools for Programmers", Datamation, pp: 131 to 144, February 1983.

Standish (1995), *Chaos*, The Standish Group, *retrieved from* http://www.standishgroup.com/chaos.html, on March 19, 1998.

StyleWriter, http://www.stylewriter-usa.com/, last accessed October 16, 2002.

Sutcliffe A.G., Maiden N.A.M., Minocha S. and Manuel D. (1998) "Supporting Scenario-based Requirements Engineering" IEEE Transactions on Software Engineering: Special issue on Scenario Management, Vol. 24 No. 12.Waterman D.A., *A Guide to Expert Systems,* Addison-Wesly, 1985.

Voyages (1996), "Unfinished Voyages, A follow up to the CHAOS Report", The Standish Group, retrieved from http://www.pm2go.com/sample_research/unfinished_voyages_1.asp, January 21, 2002.

Waterman D., *A Guide to Expert Systems*, Addison-Wesley, 1985.

Weiss M., Stetter F., *"A hierarchical blackboard architecture for distributed AI systems*", Proceedings of the 4th International Conference on Software Engineering and Knowledge Engineering, 1992, pp. 349 – 355.

Wu C.H; Lee S.J., "An object-oriented expert system for local area network design", *Proceedings of the 5th International Conference on Computing and Information*, (ICCI '93), 1993, pp. 321 –326.

AUTHORS

Joseph Kasser has been a practising systems engineer for 30 years. He is the author of "*Applying Total Quality Management to Systems Engineering*" published by Artech House. Dr. Kasser is a DSTO Associate Research Professor at the University of South Australia (UniSA). He performs research into improving the acquisition process. 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 at University of Maryland University College. There, he developed and was responsible for the Master of Software Engineering 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 systems and software engineering in the classroom and via distance education.

Professor Stephen Cook. After graduating in Electronics Engineering from the South Australian Institute of Technology in 1977, Prof Cook commenced work as a telecommunications equipment design engineer in the UK where he also completed an MSc in Computer Science at the University of Kent. On his return to Australia, he worked in the defence electronics industry until 1988 when he joined the Defence Science and Technology Organisation (DSTO). He completed a PhD in 1990, in Measurement Science and Systems Engineering at the City University London. Prof Cook joined the University of South Australia as the Foundation Professor of Systems Engineering in 1997. He has contributed to three books and has published over fifty refereed journal and conference papers. Prof Cook is a Fellow of the Institution of Electrical Engineers (UK), a Fellow of the Institution of Engineers, Australia and Past President of the Systems Engineering Society of Australia.