A Frame-Based Approach to Requirements Engineering

Stephen C. Cook PhD FIEE FIEAust and
Joseph Kasser D.Sc., C.Eng., CM
Systems Engineering and Evaluation Centre
University of South Australia
Mawson Lakes Campus
South Australia 5095
Emails: stephen.cook@unisa.edu.au
joseph.kasser@unisa.edu.au

John Asenstorfer PhD
Communications Division
Defence Science and Technology Organisation
PO Box 1500 Salisbury
South Australia 5108
Email: john.asenstorfer@dsto.defence.gov.au

Abstract. Requirements engineering is justifiably receiving increased attention at present as deficiencies in this endeavour are often cited as being responsible for poor project outcomes. The paper opens by identifying the goals for the next generation of requirements engineering support tools and postulates that requirements engineering practice and tools will require more than an incremental improvement to achieve these goals. The approach proffered is to apply expert system techniques to the problem. Through an examination of previous research we propose that a highly capable requirements engineering tool can be built based on intelligent agents operating on a bespoke frame-based knowledge representation architecture. Evidence is presented that the approach has the representational richness and reasoning power to undertake the functionality identified as being necessary to assist requirements engineering.

INTRODUCTION

It is well recognised that a major key to successful systems developments is effective Requirements Engineering (RE). Unfortunately RE is renowned for its difficulty and is rarely done well in practice. Much has been written on what constitutes a good set of requirements and the attributes needed to express them. However there remains a gap in requirements practice between the knowledge of what is needed and the formation and management of a comprehensive and consistent set of requirements for a project. The trends towards increasing system complexity and project failure intolerance put further pressure on this aspect of Systems Engineering (SE).

Another increasingly important consideration is that systems are becoming progressively open as interoperability and compatibility assume increasing importance. Furthermore, the concept of Systems of Systems (SoS) (Maier, 1996) emphasises that many systems under development are best thought of as components of super, or meta systems, further complicating requirements elicitation and management.

This paper commences by examining traditional requirements engineering processes and tools. From this baseline, the goals of advanced RE methodologies and the functions of associated support tools are derived. The body of the paper examines research into such tools and identifies a suitable knowledge representation approach that can support the functionality needed.

THE GOAL FOR REQUIREMENTS ENGINEERING TOOLS

The real goal for RE tools is that not only should they store the requirements and provide traceability but they should also:

- Minimise the probability of designing a product that is unusable.
- Maximise the probability of meeting cost and schedule estimates.
- Provide automated reasoning to process the requirements to ascertain requirement adequacy, completeness, consistency and quality.
- Represent the requirement in a format that supports direct reasoning about the system of interest and can transfer meaning to other engineering tools.

As well as storing the requirements and providing traceability to the appropriate sources and derived or refined requirements, tools must also address two areas of activity, which are separated in the current paradigm. These areas are:

- **Requirements elicitation.** The user needs must be articulated and captured in a manner that makes it possible to build a set of requirements, which when implemented results in the delivery of system that will meet the user needs as they

evolve during the construction phase (Kasser 2000a).

- **Change management and configuration control.** The changes in user needs must be tracked, and corresponding changes made to the requirements and the implementation activities, costs, schedules and resources. Constraints and other implications must be propagated through the subsystems impacted by the change.

The universe of requirements. The universe of requirements embodies a number of categories (Kasser 2000b)

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

The full set of user requirements for a system to be acquired tends to be embodied in the subset comprising the first five of the categories. Thus the tool needs to assist in identifying the capability that is really needed by providing some mechanism to disallow unwanted requirements and ensuring that the desired requirements are captured and adequately represented.

Since changes to requirements comprise ammendments, additions and deletions, the change management process can embody the requirements elicitation process. This process can be considered from several viewpoints:

- **The temporal viewpoint** – requirements change over time and that change must be managed. Moreover, the initial requirements elicitation process is itself a change from no system to the first iteration of the system (or from an existing system to the first iteration of the replacement system).
- **The management viewpoint** - since the change management process affects the work, informed decisions about the effect of change on the use of resources, schedules and costs need information that is currently contained in several different tools, including project management, cost accounting, and configuration control.
- **The feasibility viewpoint** – requirements must be feasible. This means the system must be able to be built with the available resources in spite of all the constraints placed on the project.
- **The product viewpoint** - while the customer may only be concerned with the mission needs, there may be other requirements imposed on the system for other reasons (government regulations, environmental conditions etc.). (Kasser 2000b)

MEETING THE REQUIREMENTS

When faced with the problem of meeting the customer's needs, good engineering practice dictates the use of one of two implementation choices (Kasser and Shoshany 2000)

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

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

In addition, engineers don't always reuse solutions or components that worked in the past they reinvent them or try to invent new ones. This practice may result in them turning a 'choice 1' situation into a 'choice 2' situation with corresponding budget and schedule costs while rediscovery takes place.

The way in which the solutions of the past are identified is by pattern matching. This indicates that if the tool is to optimize the (requirements for the) product it needs to contain solution domain knowledge and a mechanism with which to perform pattern matching.

There are three approaches to providing this capability:

Design to requirements – the traditional theoretical systems engineering approach.
Design to cost – a relatively modern modification.
Design to inventory – a bottom-up driven approach that emphasizes that the solution will, to the maximum extent possible, be comprised of components in the inventory or those available as Commercial-Off-The-Shelf (COTS) items. It is also the unrecognised situational approach used in problem solving. A typical example is when faced with a situation; the implementers pick an approach

based on what is to hand. This approach can range from the 'if the only tool to hand is a hammer all solutions require something to be hammered' to a well thought out approach using the optimal tool/approach identified from a set of tools/approaches. Design to inventory is also the process used when assembling systems against an architecture (Kasser 2001).

Any modern tool for RE that verifies the appropriateness of requirements (ie ensures that only requirements that can be implimented are imposed on a system) must then provide for the three approaches to providing capability mentioned above.

CONCEPT

The concept that we are pursuing in this paper is to apply expert system technology to requirements engineering. Expert systems (ES) are generally thought to comprise a knowledge base, an inference engine and an interface to humans and other systems. The first step in the conceptualisation of an ES is to consider the most appropriate method of representing the domain knowledge and from there consider an inferencing strategy. Some examples are

Specriter. (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. The domain of interest for this software was measuring instruments, but the design of the tool was generic and followed the architecture of a classical ES in that the inference engine and the knowledge base were separate. Cook (1990) makes a strong case for the adoption of a knowledge-based approach based on his experiences from an earlier implementation using the procedural programming paradigm (Cook, 1988) and a thorough literature survey of computer science at the time. Cook (1991), devotes an entire chapter to the identification of a suitable knowledge representation technique and after examining a broad range of options including rule-based systems, logic-based systems, and semantic nets, settled on frame-based systems.

Specriter elicited user input through a series of context-sensitive screens and tailored the list of questions to suit the problem at hand. Specriter contained domain knowledge from specification practices, measurement science, and equipment design and testing. It was able to use this to provide feedback on the quality of a response and had some limited consistency checking. 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.

Automated reasoning, however, has only recently been incorporated into commercial RE tools. Sutcliffe et al (1998) describe a system that reuses requirements knowledge through decision models and generic models. The same paper goes on to describe requirements validation using "frames" (see later) and the semi-automatic scenario generation for scenario driven RE. An ambitious reasoning system, using the M-Telos language, is described by Nissen and Jarke (1998) which supports goal-oriented analysis methods and deals with multiple conflicting perspectives and manages inconsistency.

The NATURE (European ESPRIT) project. (Pohl et al 1994) addressed integrated Artificial Intelligence (AI) assisted systems which coupled a reasoning system to a knowledge representation. The system described had the goal of producing a specification, whereas our interest is to generate a model of the system of interest. AI techniques have however been more often employed as an augmenting tool to assist reasoning in requirements engineering. NATURE in the broadest context addressed

- requirements domain theory,
- requirements process theory, and
- requirements knowledge representation theory.

Requirements domain theory investigated how to place requirements in the correct context, that is apply domain knowledge.

Requirements process theory was to develop RE methodologies, such as goal driven methods.

Requirements knowledge representation was to capture requirements and the domain knowledge and match appropriate contexts.

The work completed in 1995 with a CASE focus to embody the above schemes. Two AI augmentations resulting from this work are, first order logic to aid requirement deconfliction and Bayesian Belief Networks to assess the effect of human error in an operational system (Sutcliffe et al 1999).

CREWS. Cooperative Requirements Engineering with Scenarios is also an ESPRIT project. The project currently has four components, two of which focus on requirements elicitation and two on requirements validation.

Requirements elicitation components comprise: scenario-extended traceability which captures real world scenario usage of a system and links these scenarios with the system conceptual model; and authoring which supports structured text presentation of the captured scenarios and analysis of their contents.

Requirements validations are supported by: systematic scenario generation to test risks, exceptions, influencing factors and determining if enough scenarios have been captured; and validation tools which cover methods such as cooperative

animation for displaying various scenarios allowing users to discover errors in their specification.

CONCEPT FOR AN AI-BASED RE TOOL

From the above it would appear that a viable concept for an AI-assisted RE tool would be to capture knowledge using scenario chunks to articulate usage and goals. Once the requirements have been captured in a suitable knowledge representation, an executable conceptual model should be feasible, albeit not as eleborate as the CREWS validation process.

In order for the tool to adequately support the RE process with automated reasoning and a powerful and flexible knowledge capture mechanism, it must contain a knowledge representation that:

- Supports effective requirements elicitation and storage.
- Uses patterns to represent system requirements and goals, and embodies domain experience.
- Recognises patterns in requirements and performs substitutions on matches.
- Supports reasoning from multiple views about the requirements set including constraint propagation and sensitivity analyses to facilitate requirements negotiation.
- Supports conflict and inconsistency resolution.
- Interfaces to existing systems and software engineering design tools.

A KNOWLEDGE REPRESENTATION FOR REQUIREMENTS ENGINEERING

The concept of frames as a knowledge representation technique was initiated by Minsky (1975) although he gives credit for many of the concepts to Bartlett (1932). Minsky sought to represent common sense thought and wished to combine AI with psychology. Minsky asserts that to enable the performance of human mental activity, the unit of reasoning and the representation of language memory and perception should be larger and more organised than production rules or independent statement expressed in logic. He postulated that when a human encounters a new situation, we use a pattern matching process to select from memory a structure he calls a *frame*, which is a remembered framework to be adapted to fit reality by changing details as necessary. Thus Minsky's frames use the object-oriented paradigm for reasoning as well as modelling. (Russel and Norvig (1995) argue that Minsky's frames may be viewed as an extension of object-oriented programming using concepts such as, inheritance and default values (Birtwistle, Dahl et al., 1973) that predates the frames work.)

A frame can be considered to be a data structure for representing a stereotyped situation. Minsky uses examples such as being in a certain kind of room or going to a child's birthday party. Attached to each frame can be several kinds of information, for example, how to use the frame, what can happen next in the situation described, what to do if the unexpected occurs, and all manner of details about the situation under consideration. This degree of richness can enable a frame-based system to match patterns and take appropriate actions.

It is customary to follow Minsky's suggestion that the top levels of the frame should be fixed and represent things that are always true about the situation (e.g. the problem invariant in C4ISR terminology). The lower levels have *slots* that must be filled by specific instances of the data. Each slot can specify conditions its assignment must meet and may take the form of a sub-frame. Collections of related frames are linked together into frame-based systems and important actions are mirrored by transformation between the frames of a system. Thus in visual systems, the different frames of a system can represent a scene from different viewpoints and the transformation between frames corresponds to the observer moving to another viewpoint. This is analogous to the RE approach of capturing requirements from different perspectives.

The power of frames is based on the inclusion of expectations and other kinds of presumptions. These take the form of defaults placed in the frame's terminal slots. These defaults are loosely attached and can be replaced by new items that fit the current situation better. Defaults can also be generated automatically. In the information retrieval network for machine vision described by Minsky (1975), a matching frame is sought to describe the current scene. If an adequate match is not available, the network provides another frame which possesses whatever information is appropriate and generates defaults to cover values which are not explicitly known.

Winston (1993) makes it clear that frames are an extension of semantic networks. Thus all the desired properties of semantic nets such as inheritance, attached procedures, and defaults are available in frames.

THE CELLULAR PHONE EXAMPLE

When faced with the problem of designing a cellular telephone, one way of implementing the design is to base it on a Digital Radio. The pattern matching process determines that the Cellular phone is very similar to a Digital Radio, hence the capability of the Digital Radio may be useable in building a cellular phone. For example, a Digital Radio comprises:

- Antenna
- Modulator and Demodulator
- Forward Error Coder and Decoder
- Bit message Framer and Deframer
- Key Pad for tuning and driving menu
- Message/Diagnostic display
- Frequency range 3 MHz to 3 GHz

Thus a cellular phone can be considered as a Digital Radio, that is it inherits its features, however:

- Keypad is overridden or changed and used for dialling plus driving menus
- Frequency range is a subset comprising 870 MHz to 930 MHz
- Voice codec is a new component that was not present in the Digital Radio.

This simple example illustrates how the existing capability in the Digital Radio is augmented by new functionality based on the requirements to produce the cellular phone with a maximum amount of reused or inherited functionality.

This simple example also illustrates inheritance through the "is a type of" link in object oriented analysis and allows us to reason about a mobile phone. We note the mobile phone has some explicit features, note the keypad is used differently to the way it is used in the digital radio, the frequency range has greater constraint and the mobile phone has the additional feature of a voice codec. The remaining features are inherited from, and remain identical to those in, the Digital Radio. The more explicit features of a frame, over-ride the inherited features.

Hayes (1979) considers stereotypical frames to be bundles of properties expressible in predicate calculus and particular instances as simply instantiations. He notes, however, that while the meanings appear to be the same, the inferences allowed by frames, because of their structure, may be different from those sanctioned by logic. Ringland's (1988) discussion of frames, which is based on the papers of Minsky and Hayes augmented by input from later researchers, concludes that frames are more than an alternative logic representation, as many features such as the inherent structure of the frame system and the special properties of defaults, such as non-monotonicity and loose attachment are meta-logical.

Bender (1996) points out that semantic networks, and hence frames and their object oriented representations, support non-monotonic reasoning systems. He describes the strengths and limitations of default reasoning and indicates how it may be used in frames and semantic net structures. This reasoning system can be used, with extensions, to isolate and deal with contradictions, but more importantly it has the ability to localise contradictions so that they don't propagate in long chains of reasoning. It appears that this form of reasoning can not yet imitate common sense reasoning.

Winston (1993) devotes a chapter on how frames can capture commonsense knowledge although a special language of discourse is required to enter the knowledge into the frame. He also describes in some detail how constraints, both numeric and symbolic can be propagated through semantic networks. Clearly this property is of great importance to RE.

By recognising the link between object oriented design techniques and frames we can utilise some of the more advanced concepts recently developed for software engineering practice. One useful abstraction is that of frameworks. A framework addresses a specific problem domain and provides a solution by encapsulating knowledge through a collection of object classes (Birrer and Eggenschwiler, 1993). Abstractions such as frameworks will allow rapid specification of systems to meet specific scenarios. Although it may be a challenge to aid the construction of frameworks with automated reasoning, however once a framework has been constructed it will be less difficult to assist the requirements engineering with a reasoning engine. Another abstraction is that of patterns (Gamma, et al 1993). Design patterns name, identify and abstract common themes in object-oriented design. They capture intent behind designs, group objects and define the relationships and responsibilities between them. They can be considered higher level building blocks from which more complex designs can be implemented. Thus a framework consists of recurring patterns. Patterns admit an additional level of design reuse where implementations may vary for specific designs but the micro-architecture that patterns embody still apply.

Maiden et al (1998) used the concept of patterns and developed a pattern language for the validation of system requirements. Patterns are applied to scenarios, for scenario-assisted requirements processes, and requirements documents to detect deficiencies and improve the design of a system. They make the point that little has been written on patterns in requirements engineering.

The reasons for selecting frames for Specriter around a decade ago are still valid today. Frames offer good expressive power, they can be modelled in object oriented or declarative (logic) languages to support automatic reasoning, they are easier to understand and maintain than alternative representations, and they possess an inherently hierarchical structure.

(Kasser 2000c) proposed expanding the traditional Requirements Traceability Matrix 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). The information in the QSE include but are not limited to the items shown in Table 1.

ARCHITECTURE FOR A FBRET

Figure 1 illustrates the conceptual design of a Frame-Based Requirements Engineering Tool (FBRET). The external data (knowledge) bases represent various views we take on the requirements. The views we utilise are based on a generic set proposed by Hitchins (2000). The underlying knowledge representation will be capable of selecting suitable frameworks for the system of interest through determination of the domain and the scenarios of interest. The selected framework of interest will link

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.

Table 1: The FREDIE Quality System Elements (QSE).

to the domain knowledge for the various views, understand design patterns and hold the frame tree. The framework will also hold information regarding interfaces to the external world. The user will be guided through the development of a model that conceptualises the systems of interest. Once the model is completed it can be stimulated through knowledge of the external world interfaces to identify feasibility, management, and product issues that may invalidate it. Specific aspects of model are interrogated through the use of agents. These agents assist in:

- the elicitation process.
- determining the feasibility of the system by analysing constraints.
- tracking changes and implementing change control.
- providing information about affordability.
- describing capability for partial implementation using priorities, etc.
- facilitating access to domain knowledge.

An example frame structure that is to be used in the implementation of the resulting model is shown in Table 2. The rationale for the structure is as follows. The identification slot is used to construct the frame tree and uniquely identify each requirement. The default slot holds rules that will complete the slot under default conditions, allowing reasoning with the requirement, even in the event of partial knowledge. The agent slots hold information specific to the agents, as an example feasibility would consider various criteria, constraints being one, and using appropriate rules make a deduction regarding the feasibility of implementing the whole or part of the system. The Quality Systems Element slot has the parameter set as illustrated in Table 1. This information will be accessible by the various agents, in particular the report generator or interface agent that would pass information to other commercial RE tools.

Identification Slot - ID, Parent, Level
Default slot- Rules to complete requirement under default circumstances
Agent slots - Specialist slots for agents: to hold rules for feasibility, affordability, links to domain knowledge,
User interface slots - Rules and instructions for the user interface
QSEslot - slot to hold the requirements and associated elements

Table 2: An illustrative frame structure.

THE COMMUNICATIONS REQUIREMENTS EVALUATION & ASSESSMENT PROTOTYPE (CREAP)

As stated above, today's requirements engineering processes do not have very effective checks to ensure that system requirements are feasible. In the main, this is because the feasibility of requirements is governed by domain knowledge (application or technology) which is lacking in the writers of the requirements. The Communications Requirements Evaluation & Assessment Prototype (CREAP) project is being undertaken at the Systems Engineering and Evaluation Centre. It is constructing a prototype

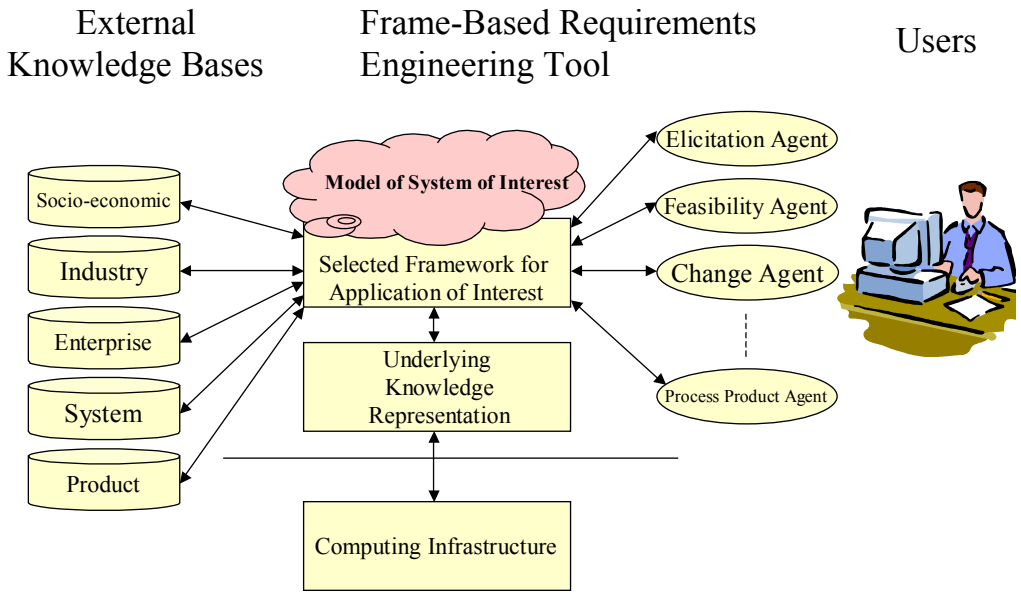

Figure 1: An architecture for a frame-based requirements engineering tool.

software package that will provide a tool for investigating the feasibility of adding artificial intelligence to requirements engineering tools.

The intial prototype will integrate military communications domain expertise and requirements engineering practice in order to ensure the feasibility of requirements imposed on communications systems in a design to inventory context. To facilitate developing the generic shell, two instances of communications models are being developed, namely

- an HF communications model
- a satellite communications model

CONCLUSION

Automated approaches to RE providing features such as reasoning, structured knowledge capture, generic design solutions and elicitation assistance are becoming increasingly important as the systems and the environments that they interact with become increasingly complex. A FBRET constructed along the lines of the architecture above could undertake many of the important functions of RE including

- (Semi-)automatic requirements elicitation
- Reasoning
  - consistency
  - completeness
  - containing systems requirements inheritance (from an overarching enterprise technical architecture and international standards)
  - design rules feasibility checking (from design tools)
  - requirements negotiation
  - cost
- Impact assessment of proposed changes
- Change management
- Sensitivity analysis
- Report generation

The architecture lends itself to direct interoperability with systems engineering and software engineering tools at all scales of complexity. Furthermore, the tool could learn by recording the requirements practice used within an enterprise and hence provide greater assistance over time.

The tool proposed in the paper provides an environment that constructs a model of the system of interest. This model can be interrogated and parameters altered to determine whether the requirements can still be met. The model approach enhances the elicitation process as feed back to the stakeholders can be made more meaningful.

The CREAP system is currently under construction and will investigate the applicability of a number of the core concepts presented herein.

REFERENCES

Bartlett F.C. (1932). Remembering, Cambridge University Press, Cambridge.

Bender E.A. (1996), *Mathematical Methods in Artificial Intelligence,* IEEE Computer Society Press, Los Alamitos, California, 1996.

Birrer A. and Eggenschwiler T. (1993), “Frameworks in the Financial Engineering Domain: An

Experience Report." European Conference on Object-Oriented Programming, 1993 pp 21 - 35.

Birtwistle G.M., Dahl O.-J., Myhrhaug B. and. Nygaard K. (1973), *SIMULA BEGIN*, Petrocelli/Charter, New York, 1973.

Cook S.C. (1998), "Automatic Generation of Measuring Instrument Specifications", *Measurement*, Vol. 6, No. 4, pp155-160, 1988.

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.

Gamma E., Helm R., Johnson R. and Vlissades J., (1993), "Design patterns: Abstraction and Reuse of Object-Oriented Design." European Conference on Object-Oriented Programming, 1993, pp 406 – 431.

Hayes F. (1979) "The Logic of Frames", in Frame Conceptions and Text Understanding, ed. Metzing D., Walter de Gruyter & Co, Berlin.

Hitchins D. (2000), "World Class Systems Engineering – the 5-layer Model", www.hitchins.co.uk/5layer.html

Kasser (2000a), "The WebConference: A Case Study", *The INCOSE - Mid-Atlantic Regional Conference*, Reston, VA, 2000

Kasser J.E., (2000b) "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 (SETE 2000),* Brisbane, Australia.

Kasser J.E. (2000c) "A Framework for Requirements Engineering in a Digital Integrated Environment", *the Systems Engineering Test and Evaluation Conference (SETE 2000).*

Kasser J.E. and Shoshany S., (2000), "Systems Engineers are from Mars, Software Engineers are from Venus", 13th International Conference Software & Systems Engineering and their Applications (ICSSEA '2000), Paris, France, December 5-8, 2000.

Kasser J.E. (2001), "Writing Requirements for Flexible Systems," INCOSE UK Spring Symposium, 2001.

Maiden N.A.M., Cisse M., Perez H. and Manuel D. "CREWS Validation Frames: Patterns for Validating Systems Requirements", Fourth International Workshop on Requirements Engineering: Foundations for Software Quality (RESFQ), Pisa, Italy, June 8-9th, 1998

Maier M.W., "Architecting Principals for Systems-of-Systems", *INCOSE Annual Symposium*, 1996.

Minsky M. (1975). "A framework for representing knowledge." The Psychology of Computer Vision, Ed. Winston P.H., McGraw-Hill, New York, p211-277.

Nissen H.W. and Jarke M. (1999) "Repository Support for Multi-Perspective Requirements Engineering", Special Issue on Meta Modelling and Method Engineering, Information Systems, Vol 24, No. 2.

Pohl K., Assenova P., Doemges R., Johannesson P., Maiden N., Plihon V., Schmitt, J.-R. and Spanoudakis G. (1994) "Applying AI Techniques to Requirements Engineering: The NATURE Prototype", ICSE – Workshop on Research Issues in the Intersection Between Software Engineering and Artificial Intelligence, Sorrento, Italy, May 1994.

Ringland G. (1988). "Structured Object Representation - Schemata and Frames." In Approaches to Knowledge Representation, Ed. Ringland G.A. & Duce D.A., Research Studies Press, Letchworth.

Russell S.J. and Norvig P. (1995), *Artificial Intelligence: A modern Approach, Prentice Hall,* 1995.

Sutcliffe, A., Galliers J. and Minocha S. (1999) "Human Errors and System Requirements" IEEE International Symposium on Requirements Engineering, pp 23 – 30.

Sutcliffe A.G., Maiden N.A.M., Minocha S. and Manuel D. (1998) "Supporting Scenario-based Requirements Engineering" IEEE Trans on Software Engineering: Special issue on Scenario Management, Vol. 24 No. 12.

Winston P.H. (1993), *Artificial Intelligence,* 3rd Edition, Addison Wesley, 1993.

BIOGRAPHIES

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 President of the Systems Engineering Society of Australia.

Joseph Kasser D.Sc. C.Eng, CM, has been a practicing systems engineer for 30 years. He is the author of "*Applying Total Quality Management to Systems Engineering*" published by Artech House. 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. 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.

Dr John Asenstorfer. Dr Asenstorfer graduated from the University of Adelaide in Physics, Maths and Engineering in 1980. He was employed at BHP Whyalla as maintenance and research engineer, which gave broad experience in control systems, electric distribution and practical electric/electronic system analysis. In 1983 John joined the South Australian Institute of Technology and taught in microprocessor and electonic systems. At this time he began research into communications systems, in particular forward error control, channel modelling and information theory. In 1986 he moved to the Adelaide campus of the SAIT (now University of South Australia) and became one of the inaugural members of the Institute for Telecommunications Research. In 1987 he completed a Masters degree by research in communications and in 1995 a PhD. At that time he was the Director of the Mobile Communications Group within the Institute, undertaking research in multi-user CDMA, multiple access schemes and mobile network access protocols. In 1999 he joined DSTO and currently heads a group investigating network architectures. He remains an adjunct associate professor of the University of South Australia.