Researching the Properties of Object-Oriented Requirements

Joseph E. Kasser
Systems Engineering and Evaluation Centre
University of South Australia
Mawson Lakes, SA. 5095
Joseph.kasser@unisa.edu.au

Abstract

This paper summarises the progress of a research effort into the properties of object-oriented requirements from the perspective of Total Quality Management (TQM) and a starting assumption that since requirements drive the work of implementing the system, object-oriented requirements have both the traditional properties describing the product but also have properties than can assist in ensuring a cost-effective production process. Several candidate properties that have been identified and their usefulness are then discussed. Some concept demonstration tools that have been developed are summarised, and the paper concludes by pointing out that the object-oriented properties can be used to provide views of a proposed system at the System Requirements Review that would be difficult to observe in the current paradigm.

Object-Oriented Systems Engineering

Several approaches to Object-Oriented systems engineering have been proposed (e.g. Hopkins 1998; Meilich 1999; Lykins 2000) and there is an INCOSE working group working on upgrading the UML to add "systems" aspects. However, current thinking in Object-Oriented systems engineering has a disconnection in its process as shown in Figure 1. This is because while concepts of

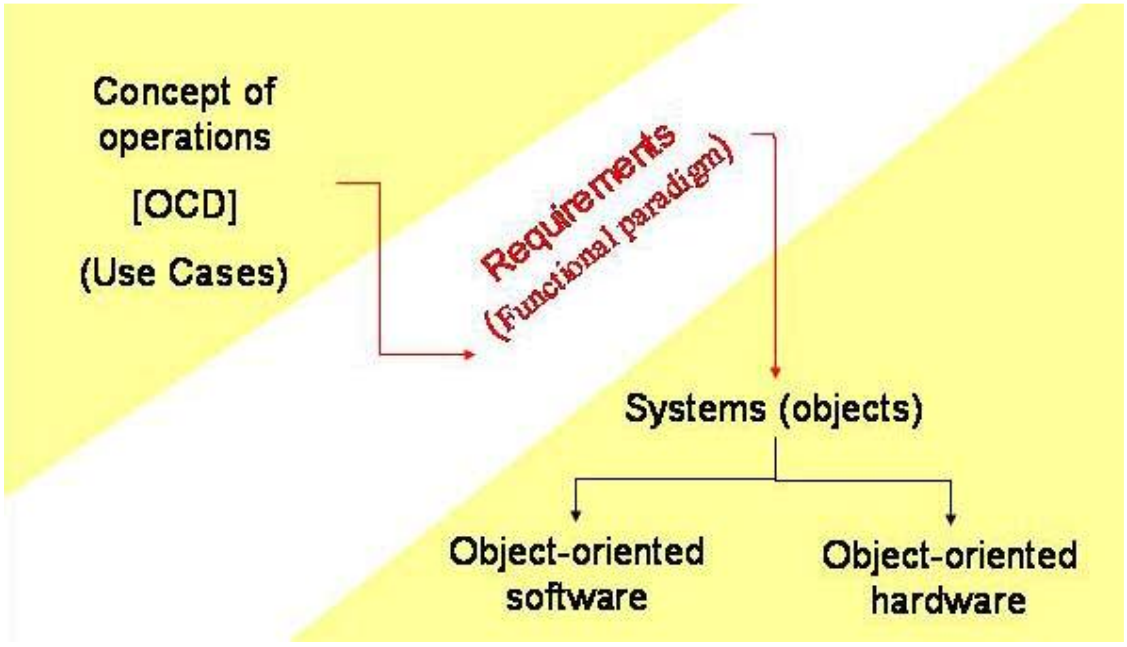

Figure 1 Object-Oriented Systems Engineering

operations are stated in an object-oriented manner in the form of Use Cases involving the interaction of objects, and the system developed consists of objects, requirements remain firmly in the functional paradigm.

There has been recent recognition that a requirement is more than just the imperative statement. For example, both Alexander and Stevens (2002) and Hull et al. (2002) discuss additional properties of text-based requirements (e.g. priority and traceability) in conjunction with improving the writing of requirements. However, in practice, while some requirements management tools provide the capability for adding these properties, in general, there is difficulty in adding these additional properties to the traditional requirement document or database and then managing them. The object-oriented approach of encapsulating data with functionality can be used to incorporate these and other properties of requirements (e.g. risk) by definition. However, in order to apply the object-oriented approach to requirements in an optimal manner, the question to be researched is "what are the properties of an Object-Oriented requirement? The answer is not simple.

Before attempting to identify the properties of object-oriented requirements, a set of rules were established based on the maxim that a good requirement has the following three characteristics:

1. **It describes something about the physical system that will meet the needs of the customer.** This is the traditional text based sentence that covers the functional and non-functional aspects of the system being produced.
2. **It facilitates (or rather not does not impede) the production process**. This characteristic is derived from Total Quality Management (TQM) (defined by NASA (1992) as the application of systems engineering to the work environment) and is concerned with the effectiveness of the production process. While requirements define a need, they can also be viewed from the contractual perspective. The cost of realising a system is based on the work and materials needed to transform needs into systems. Properties based on this characteristic include vagueness and ambiguity, namely properties that lead to cost escalations, schedule delays, or the provision of undesired functionality.
3. **It is something the customer really wants.** This is the most difficult characteristic since customers do not always to state the requirement.

Requirements drive the work

The requirements elicitation process produces a set of requirements, which represent the documentation of a function and performance of a system that meets the customer's needs. Consequently, every element of the work ought to be traceable (and chargeable) to a specific requirement or set of requirements. For example, the United States Military Standard MIL-STD-2167 prescribed software development folders (SDF). The Standard states that the set of SDFs shall include (directly or by reference) the following information:

- Design considerations and constraints
- Design documentation and data
- Schedule and status information
- Test requirements and responsibilities
- Test cases, procedures, and results

The work to implement a set of requirements (realise a system specified by the requirements) takes place in three streams of activities (management, test and evaluation (quality), and development) (Kasser 1995).

The SDF contains information pertaining to the three stream of work.
Every requirement may be thought of as having properties driving the work in each of the three streams as well as the traditional properties that describe something about the product (capability) to be built. This concept produces a view of a requirement statement as the tip of an iceberg, where the statement can be seen, but the underlying work to produce the capability that meets the requirement is hidden. An alternative view in the form of a documentation tree is presented in Figure 2.
Early research into the properties of object-oriented requirements based on TQM, and a review of the requirements engineering and systems engineering literature, has identified:

- a set of Quality System Elements (QSE) that form candidates for properties of an object-oriented requirement (Kasser 2003);
- some functionality that can be incorporated into the requirements object (Kasser 2003); and
- a set of requirements for writing the wording of requirements (Kasser 1995).

In order to investigate and determine the usefulness of these candidate properties and functionality, a set of prototype educational tools for teaching systems and software engineering (PETS) have been developed and used in postgraduate classrooms.
Effectively, object-oriented requirements engineering and management not only performs the requirements engineering at the front end of the System Life Cycle (SLC), but also provides integrated information for the functions of project management, design, development, and test and evaluation, as performed in the current paradigm. As in the current paradigm, the implementation work plan can be published in three documents, namely:

- The system requirements document;
- The systems engineering test and evaluation plan;
- The system engineering program management plan.

Kasser (2000) researched the SLC from an

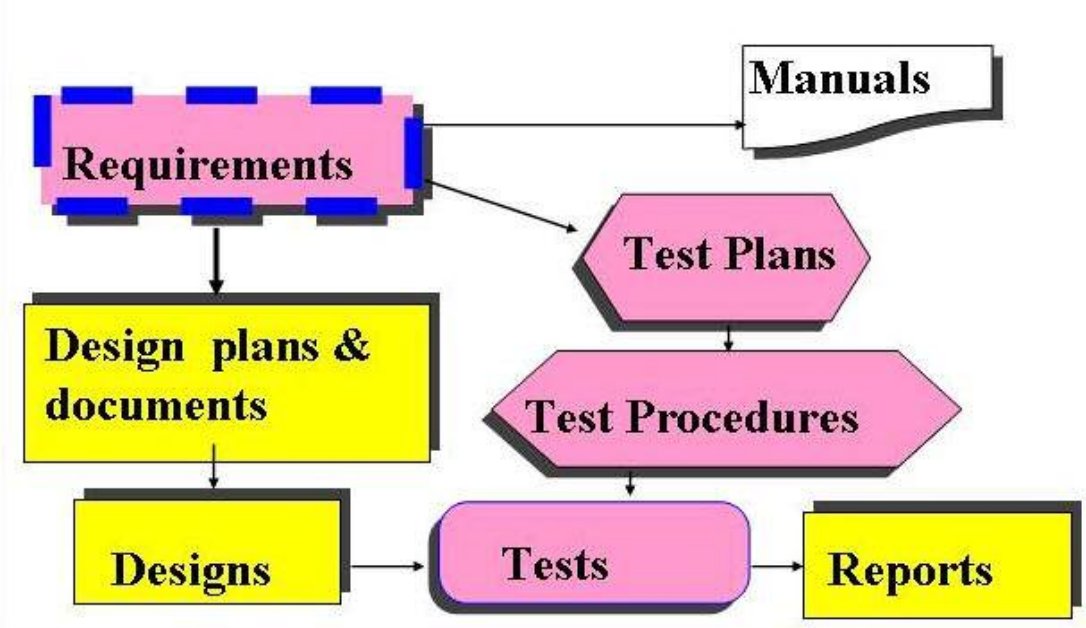

Figure 2 Requirements Drive the Work

information perspective, and determined that these documents may be considered as summaries of the properties of the object-oriented requirements in each of the appropriate stream of work. Consider the contents of each of the documents. For all documents, the following properties of the requirement apply.

- The unique identification number – to clearly identify the requirement.
- The text of the requirement statement – conforming to the requirements for writing requirements (Kasser 1995).

The other properties of the requirements are document specific as follows:

The system requirements document

The system requirements document contains the documented solution of what the proposed systems' function and performance must be to provide a solution to the customer's problem (based on the concept of operations). The system requirement document should contain the following information for each requirement:

- **Traceability to source(s)** – i.e. the concept of operations, regulations, people, etc.
- **Rationale for requirement** – to communicate the reason why the requirement was included in the first place. This information is important for considering change requests during the operations and maintenance phase of the system's life cycle. This information is sometimes included as comments in the current paradigm, but is not required.
- **Traceability sideways to other documents** (or databases) at the same level of decomposition of the system. This provides information for use by the configuration control board in considering the impact of requested changes.

The systems engineering test plan

The systems engineering test plan drives the testing and evaluation process. The systems engineering test plan should contain the following information for each requirement:

- **Acceptance criteria** – which are provided in response to the question "How will we know that the requirement has been met by the system?"
- **Planned verification methodology(s)** - demonstration, analysis, etc. Everything cannot be tested to determine compliance. For example, how do you test a fuse? If it blows at the correct current rating, it passed the test, but is then useless.
- **Testing parameters** – the sections of the test plans and procedures that verify the system meets the requirement.
- **Resources needed for the tests** – people, equipment, time, etc.

The systems engineering program management plan

The systems engineering management plan contains the planned resources and schedule necessary to perform the design and testing activities. The systems engineering management plan should contain the following information for each requirement:

- **Traceability to implementation** – identifies the Build in which the requirement is scheduled to be implemented.
- **The priority of the requirement.**
- **The estimated cost** to construct and test the elements of the system that provided the functionality specified by the requirement.
- **The level of confidence in the cost estimate**.
- **Risks** – probability and severity of implementation, programmatic, and any other identified, and their mitigation.

- **Production parameters** – the Work Breakdown Structure (WBS) for the work to be done to meet the requirement.
- **Required resources** for the work.

Kasser (2000) summarised this information in the form of a set of QSE as being necessary for effective system and software development. A suite of prototype software agents (Kasser et al, 2003) have been developed together with a candidate object-oriented requirement database schema (Kasser 2003) and used in an educational environment.

The tool concept

The QSE are not new. They are known and have been used in project management and systems engineering for many years (eg. MIL-STD-2167A as described above). Some of them have also been incorporated as fields in requirements management tools from time to time. However, these instances seem to be the exception rather than the rule. The QSE also do not seem to have been used together in an integrated manner. The suite of tools provide the capability to use the QSE in an integrated manner, in a way that could be widely adopted, thus changing the paradigm.

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 (2000) also described the concept of a suite of tools for accessing the information in the QSE. Kasser and Cook (2003) discussed the projected implementation of the suite using a rapid incremental solution construction approach, which maps into both the Blackboard (McManus 1992) and Agent based approaches.

The prototype educational tools for systems and software (PETS) engineering

A suite of prototype tools with a similar user interface is being developed and early prototypes have been deployed in an educational setting. The plan is for the tools to continue to evolve into intelligent agents (applying the object-oriented concept of placing methods (processes or functionality) into the object) using the Blackboard approach as more is learnt about their use. These initial tools are:

- The First Requirements Elucidator Demonstrator (FRED).
- Tool to InGest and Elucidate Requirements (TIGER).
- Acceptance Criteria Elucidator (ACE).
- Requirement Enhancing documentation Tool (ET).
- Requirements completeness MAXimiser (MAX).
- Requirement priority documentation tool (RP).
- Risk documentation And profiling Tool (RAT).

Each of the tools is outlined below.

The First Requirements Elucidator Demonstrator (FRED).

An early prototype of FRED was presented by Kasser (2002) as a tool that ingested requirements from documents and identified potential defects in requirements by parsing the text for the presence of a set of “poor words”. FRED performed a syntax check on the text of the requirement pointing out potential defects. This is a similar function to the way in which a word processor grammar and style checker is used, the tool points out that there may be a defect, and it is up to the user to determine if the defect is there and how best to correct it. FRED was based on automating and improving the manual process performed during a Requirements

ess performed during a Requirements Workshop held in courses leading to the Master of Software Engineering degree at the Graduate School of Management and Technology at University of Maryland University College. FRED produced a Figure of Merit (FOM) for the document. The FOM is a simple one-dimensional measurement for the quality of a document based on the presence or absence of "poor words". The FOM allows comparisons to be made of the quality of documents of different sizes. The FOM was calculated using the formula

```
FOM = 100 * (1 - number of
defects / number of require-
ments).
```

A document without any defects can achieve a FOM of 100. A document with a large number of defects can receive a negative FOM. This situation arises if the requirements in the document contain more than one defect.

Tool to InGest and Elucidate Requirements (TIGER)

FRED had no storage facilities for requirements. Results using FRED were promising so TIGER was developed to replace FRED and serve as the foundation tool for ingesting existing documents into the QSE and improving the wording of the text of the requirements. The main user interface screen for TIGER is shown in Figure 3. TIGER performs the following functions:

- Ingesting of requirements from text documents/databases and the keyboard.
- Modifying existing requirements.
- Elucidating requirements based on a set of "poor words" and pointing out up to six types of (potential) defects in each of the requirements (multiple requirement in a paragraph, possible multiple requirement, unverifiable requirement, use of "will" or "must" instead of "shall", and the existence of a user defined "poor word").
- Allowing for additional "poor words" to be added as they are identified.
- Allowing for "poor words" to be used in a requirement when their use is appropriate. For example, the requirement that "the system shall display the combined total of A and B" is a good requirement.
- Providing a built-in agent using deterministic grammar for the engineering of requirements (BADGER) that facilitates the correct format for writing requirements by, prohibiting many "poor words", and minimizing the need for retyping by the use of drop down lists (Scott 2004).
- Producing a report documenting each occurrence of a "poor word" in the requirements.
- Producing the FRED Figure of Merit.

Acceptance Criteria Elucidator (ACE)

ACE allows the user to create, view, add, and modify acceptance criteria for requirements in the QSE database. ACE also provides a printout of each requirement in the database together with its acceptance criteria.

Requirement Enhancing documentation Tool (ET)

ET provides for the viewing, addition, modifying, and reporting, of the following QSE elements

- Keywords;
- Rationale for the requirement;
- Traceablity to source(s) and sideways.

Requirements completeness MAXimiser (MAX)

Many requirements documents are written for systems that are similar to, or are subsets of, existing systems. MAX employs the concept of inheritance and allows the user to

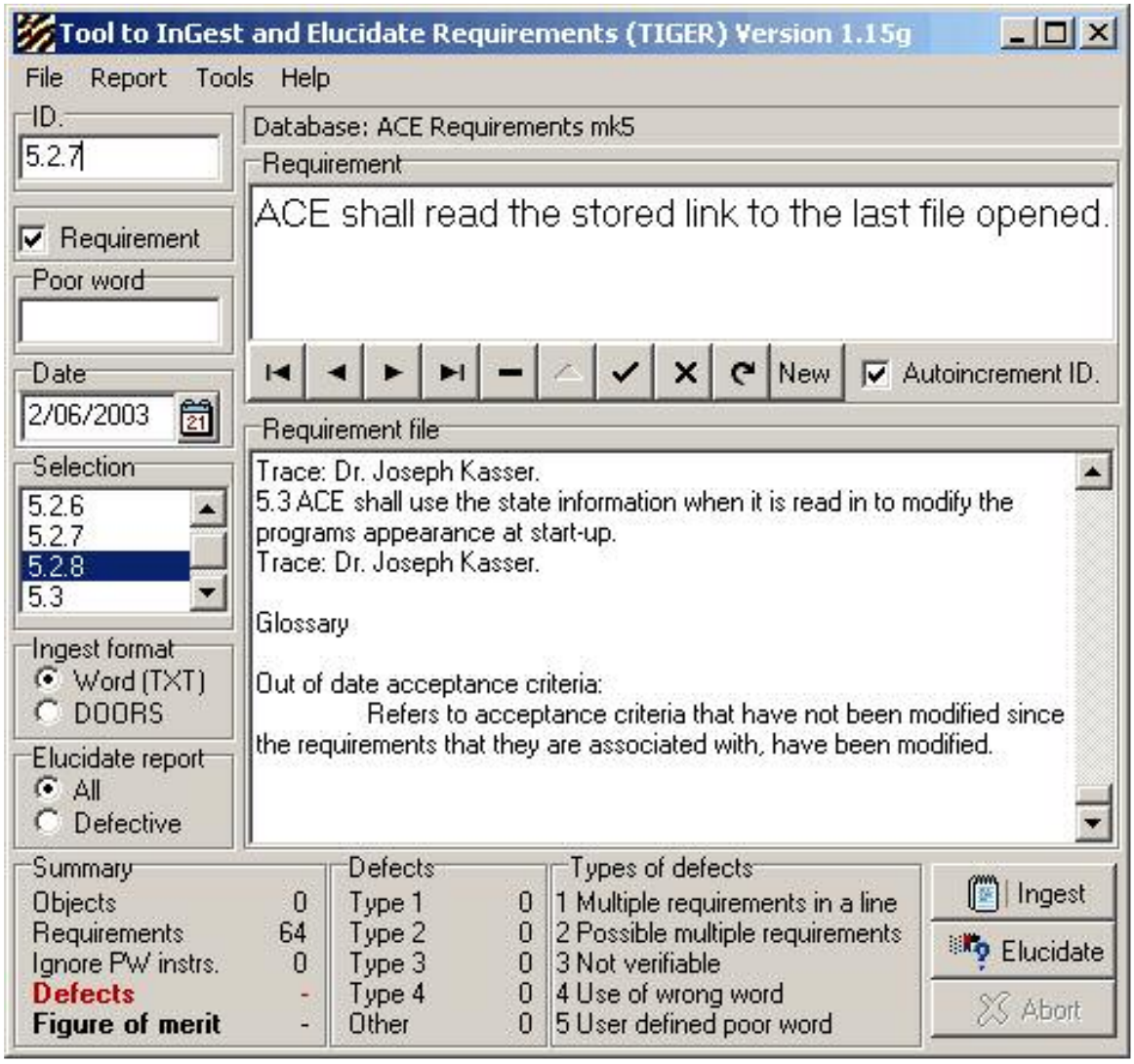

Figure 3 TIGER

inherit requirements from an existing QSE or requirements database. For example:

- Some of the requirements for one PET can be inherited from a previous tool database (e.g. user interface requirements).
- When developing requirements for a specific type of system (e.g. communications satellite), non-functional requirements for the object class of communications satellites can be inherited (e.g. thermal vacuum, humidity, and vibration) hence maximizing the completeness of the requirements for the specific instance being constructed.

MAX also provides the capability to add, view, and modify requirements.

Requirement priority documentation tool (RP)

This tool allows the user to view, document, and modify, an assigned priority (1-10) to each requirement. The tool also shows the total priority profile as a Pareto chart identifying the number of requirements in each range of priority.

Risk documentation And profiling Tool (RAT)

This tool allows the user to view, document and modify:

- The probability and severity of an identified risk.
- The rationale for the risk.
- The risk mitigation strategy.

The tool also displays the total probability, severity and risk as a Pareto chart.

Early results

Two kinds of results have shown up during the early stages of the research:

- An improvement in the teaching of requirements engineering;
- The provision of information at the time of the System Requirements Review (SRR) that is unavailable in the current paradigm.

An improvement in the teaching of requirements engineering;

TIGER was the first tool to be used in a class lecture on requirements engineering in three postgraduate courses. Before TIGER was introduced, the discussions in the tutorials focussed on the structure and format of requirements. After TIGER was introduced and used to elucidate sample requirements, the focus of the in-class discussions changed to cover the difficulties of writing good requirements. This is a significant shift in perspective (Kasser et al., 2003).

The provision of information at the time of the System Requirements Review that is unavailable in the current paradigm

The initial set of tools when used individually or in combination provide some interesting views of the system, namely:

- TIGER provides a FOM of the text of the requirements. The assumption is that the lower the FOM, the greater the probability of cost and schedule overruns due to poorly worded requirements.
- The use of ACE to document the acceptance criteria results in better requirements. The dialog as a result of the question "how will we know if or when the requirement is met?" clarifies the intent of the need statement; indeed sometimes resulting is rewording of the requirements statement. Thus ACE helps meet the third characteristic of a good requirement, namely is it something the user really wants.
  The first combination of ACE and TIGER in the postgraduate classroom in February 2004 resulted in a marked improvement in the way the students discussed requirements and elucidated the wording. At the end of the exercise the students demanded both an elucidation function for acceptance criteria in the way that TIGER elucidated requirements, and guidelines for writing good acceptance criteria. Until then the concept of acceptance criteria had been glossed over
- RAT provides the functionality to access risk properties including mitigation strategies. This functionality provides for the Risk Plan to be associated directly with the requirements. The Risk Plan in this paradigm is a printout (abstracted view) of the risk properties of the requirements in the database.
  RAT also provides risk profiles of the system at the SRR. This allows us to ask the question "what is the risk profile of a system of this type?" Risk profiles for new systems that are copies of existing systems should be low. Risk profiles for systems breaking into new territory would be different. If the risk profile of a proposed system doesn't match the risk

profile for its category of systems, then questions need to be asked, starting with "why?"

- Documenting the priority of the requirements allows the customer to be sure that the high priority requirements have been assigned to early Builds to ensure that the most desired functionality is produced in the event that funding is reduced, or the project runs out of funds. Documenting the priority serves the same purpose for the high priority requirements.
- The risk, priority, and cost estimate properties of a requirement can be easily examined and discussed at the SRR (before any resources are expended in the implementation phase). For example, are requirements with the following combined properties really needed: (1) high-risk low-priority; (2) high-cost low-priority; (3) high-risk high-cost? The answers may be in the affirmative, but at least the decision to implement will be an informed one.

These views and the issues they raise can help prevent the waste of resources and increase the probability that system as implemented complies with its real requirement, namely is a Quality system (Crosby 1979).

Conclusion

Early research into the properties of object-oriented requirements based on TQM, and a review of the requirements engineering and systems engineering literature, has identified a set of candidate properties and produced a set of conceptual tool demonstrators that have the potential to significantly improve the current systems development process.

Postscript

The tools which operate on personal computers using the Microsoft Windows (2000 and XP) operating system, together with some teaching materials in the form of PowerPoint presentations are freely available from the author.

Further research into these tools and development of the additional ones (Kasser et al., 2003) is in progress.

References

Alexander, I. F., Stevens, S., *Writing Better Requirements*, Addison-Wesley, 2002.

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

Deming, W.E. 1986, Out of the Crisis, MIT Center for Advanced Engineering Study, p11.

Glover, S.J., Bennett, K.H. (1996). "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.

Hopkins, F.W., Rhoads, R.P. "Object Oriented Systems Engineering – An Approach", *Proceedings of the 8th International INCOSE Symposium*, 1998.

Hull, M.E.C, Jackson, K., Dick, A.J.J., *Requirements Engineering*, Springer, 2002.

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

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

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

Kasser, J.E., Cook, S.C. (2003). "Using a Rapid Incremental Solution Construction Approach to Maximize the Completeness and Correctness of a Set of Requirements for a System", *Proceedings of the 13th International Symposium of the INCOSE,* Washington D.C.

Kasser J.E., Tran X-L, Matisons S., "Prototype Educational Tools for Systems and Software (PETS) Engineering", *Proceedings of the 14th Annual Conference for Australian Engineering Education*, Melbourne, 2003.

Lykins, H., Friedenthal, S., Meilich, A., "Adapting UML for an Object Oriented Systems Engineering Method (OOSEM)", *Proceedings of the 10th International INCOSE Symposium*, 2000.

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.

Meilich, A., Rickels, M., "An Application of Object Oriented Systems Engineering (OOSE) To an Army Command and Control System: A New Approach to Integration of System and Software Requirements and Design", *Proceedings of the 9th International INCOSE Symposium*, 1999.

Military Standard 2167A. Defense System Software, 29 February 1988.

NASA Systems Engineering Handbook, draft, September 1992.

Scott. W., (2004). "Application Of Context-Free Grammar To Improve Requirement Elicitation", to be submitted to the *SETE 2004 Conference,* Adelaide, Australia.