Model-Based Systems Engineering: Back to the future?

Joseph Kasser
Temasek Defence Systems Institute
National University of Singapore
Block E1, #05-05, 1 Engineering Drive 2 Singapore 117576
tdskj@nus.edu.sg

Abstract. According to some, Model-Based Systems Engineering (MBSE) is, is about to, or will become, the status quo of systems engineering. Jenkins wrote that one of the roles of the systems engineer was to challenge the status quo but few systems engineers have actually challenged the status quo of systems engineering. So, in that spirit, this paper presents some perceptions of MBSE from the holistic thinking perspectives (HTP) (Kasser, 2013b). The paper shows that:

1. MBSE is a laudable attempt by systems engineers in the B paradigm of systems engineering (Kasser, 2013a) pages 439-455) to return to the original A paradigm.
2. MBSE lacks the awareness of how current information technology could smarten up systems engineering tools.
3. Past research at the Systems Engineering and Evaluation Centre (SEEC) at the University of South Australia (UniSA) is among the research that has produced functionality that could and should be introduced into the next generation of systems engineering tools.

Key words: Model based systems engineering.

1. Introduction

Jenkins wrote that one of the roles of the systems engineer was to challenge the status quo (Jenkins, 1969). Yet few systems engineers have actually challenged the status quo of systems engineering. So, in that spirit, this paper presents some perceptions of MBSE from some of the HTPs on the perspectives perimeter (Kasser, 2013b) pages 88-116). Since the HTPs and realization that systems engineering is performed in two different paradigms, A and B, are the findings from recent research, the paper begins with a short summary of the HTPs and the two systems engineering paradigms. Section 3 then provides some perceptions of MBSE from a number of the HTPs while Section 4 applies some critical thinking and summarizes some of the research performed at the Systems Engineering and Research Centre (SEEC) in the University of South Australia (UniSA) that, together with suitable findings from other research programs, could and should be applied in the development of third and fourth generation computer enhanced systems engineering (CESE) tools with smart capabilities.

2. Big Picture Perspective

This section provides a short summary of the HTPs and the A and B paradigms of systems engineering.

2.1. The Holistic Thinking Perspectives

The holistic thinking perspectives (HTP) (Kasser, 2013b) pages 88-116) comprise a set of viewpoints on the perspectives perimeter which can be used to provide anchor points for thinking and communicating in a systemic and systematic manner. These viewpoints go beyond combining analysis (internal views) and systems thinking (external views) by adding quantitative and progressive (temporal, generic and continuum) viewpoints. The internal and external perspectives help understand a situation, but determining an optimal solution requires additional perspectives. The nine HTP's are grouped as follows:

2.1.1. External Perspectives

The external perspectives are:

1. ***Big Picture:*** perceptions about the context of the system.
2. ***Operational:*** perceptions about what the system does.

2.1.2. Internal Perspectives

The internal perspectives are:

3. ***Functional:*** perceptions about what functions the system performs and how it does them.
4. ***Structural:*** perceptions about how the system is constructed and organised.

2.1.3. Progressive Perspectives

The progressive perspectives:

5. ***Generic***: perceptions about the system as an instance of a class of similar systems.
6. ***Continuum:*** perceptions about the system as but one of many alternatives.
7. ***Temporal:*** perceptions about the past, present and future of the system.

2.1.4. Other Perspectives

The other perspectives are:

8. ***Quantitative:*** the numeric and other quantitative information associated with the system.
9. ***Scientific:*** the hypothesis or guess about the issue.

2.2. The Two Systems Engineering Paradigms

Research into the nature of systems engineering has shown that in the last 60 years, systems engineering has evolved into two paradigms identified as A and B (Kasser, 2013a) pages 439-455) as summarized herein.

2.2.1. The A Paradigm

The A paradigm is the original systems engineering paradigm which begins with a focus on converting a problematic or undesirable situation to a FCFDS and creating the Concept of Operations (CONOPS)[1]. The skills needed to do this go beyond the systems thinking because while systems thinking helps to understand the situation, it is the perceptions from the Generic and Continuum HTPs that help to identify the problem and the FCFDS. Examples of the A paradigm CONOPS include the 'to be' view in Business Process Reengineering and the conceptual model in Checkland's Soft Systems Methodology (Checkland and Scholes, 1990).

2.2.2. The B Paradigm

The B paradigm begins the System Development Process (SDP) with the collection of requirements and may or rmay not develop a CONOPS. This paradigm is inherently flawed[2] and modifications have been proposed, for example:

- Sutcliffe at al. proposed reducing human error in producing requirements by analyzing requirements using an approach of creating scenarios as threads of behaviour through a *use case*, and adopting an object-oriented approach (Sutcliffe, et al., 1999); namely they proposed a return to the A paradigm.
- Daniels et al. point out that standalone requirements make it difficult for people to understand the

[1] The FCFDS describes the solution system (Functional perspective), the CONOPS describes the context and environment in which the FCFDS will operate and how that operation is anticipated to occur (Big picture and Operational perspectives). Depending on the situation, an FCFDS may be associated with several CONOPS or several FCFDS may be associated with a CONOPS.

[2] This is because even if systems and software engineers could write perfectly good requirements, they still cannot determine if the requirements and associated information are correct and complete because there is no reference for comparison to test the completeness. Consequently, efforts expended on producing better requirements have not, and will not, alleviate the situation. The situation cannot be alleviated because the situation is akin to participating in Deming's red bead experiment, which demonstrates that errors caused by workers operating in a process are caused by the system rather than the fault of the workers (Deming, 1993) page 158).

context and dependencies among the requirements, especially for large systems and suggest using use cases to define scenarios (Daniels, et al., 2005).

- The B paradigm then uses the requirements to develop the conceptual models (Guo, 2010); (Roberts, 2008) cited in (Frittman and Edson, 2010). Frittman and Edson also wrote that "*a recent survey showed that 1/3 of all programs queried did not have a CONOPs (Roberts, 2008) page 39). Similarly, in a series of interviews and surveys conducted for this research, the majority of respondents indicated that a CONOPS was "critical" to the system's success, but was underutilized. Comparable studies on CONOPS have pointed out that even when a CONOPS is written it is often after the system is developed and done so in an effort to satisfy a Milestone Decision requirement; this "box-checking" activity strips the CONOPS of its intended role in the creative process (Nelson, 2007) pages 5-6)".* (Frittman and Edson, 2010).

3. Perceptions of MBSE from the HTPs

This section provides perceptions of MBSE from the following HTPs.

- Structrual.
- Temporal.
- Operational.

3.1. Structural Perspective

Perceptions from the Structural perspective provide a number of different definitions of MBSE, including:

- "*MBSE is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases*" (INCOSE, 2007) page 15).
- "*MBSE is the formalized application of modeling to support system requirements, design, analysis, verification and validation, beginning in the conceptual design phase and continuing throughout development and later life cycle phases*" (Friedenthal, et al., 2007).
- "*MBSE is fundamentally a thought process. It provides the framework to allow the systems engineering team to be effective and consistent right from the start of any project*" (Long and Scott, 2011) page 65).
- "*MBSE is a Systems Engineering paradigm that emphasizes the application of rigorous visual modeling principles and best practices to Systems Engineering activities throughout the [System Development Life Cycle] SDLC*" (MBSE, 2011).

3.2. Temporal Perspective

Before the personal computer became ubiquitous, systems engineers working in one of the phases of the SDP produced documents which became inputs to the subsequent phase or phases. For example in the A paradigm of systems engineering, the CONOPS document was an input to the requirements phase; the requirements document was an input to the design phase and so on. These documents were typewritten on paper originals. Copies for distribution were made using appropriate duplication technology and version control was performed using configuration management. As technology advanced, in the latter years of the 20th century, information technology provided word processors, databases, spread sheets and electronic storage. Even with electronic storage capability the paradigm hardly changed; information was still stored as separate documents and copies were printed as needed. Pioneers of the use of information technology realized that documents could be considered not as information per sé, but as displays or views of information in an underlying database. This paradigm shift is changing the storage of project information from separate documents or files to interlinked files where information is stored in one place, linked to information in another place and viewed from different perspectives. The concept behind the United States Department of Defense (DoD) Architecture Framework (DODAF) (DoDAF, 2004) is but one such example.

First generation project management tools allowed schedules, cost estimates and other project information to be stored in databases and automated the process of producing charts and reports. Similarly first generation CESE tools provided storage for product or the system-to-be-realized information and some useful functionality. For example, the tools allowed the user to link requirements to sources providing traceability. Second generation computer aided software engineering (CASE) and CESE

tools allowed systems engineers to build executable models of proposed conceptual systems as well as software-based models and simulations. MBSE is but one way of applying CESE tools.

3.3. Operational Perspective

Perceptions from the Operational perspective include:

1. MBSE is characterised by different interpretations of the word 'model'.

Consider these findings.

3.3.1. MBSE is Characterised by Different Interpretations of the Word 'Model'

MBSE meetings and workshops are characterized by people talking past each other and not communicating[3]. MBSE is characterised by different interpretations of the word 'model' as can be seen in the following examples:

- To some people a model is a way of expressing knowledge in an abstract way, yet exact, without showing unnecessary details. Others such as software engineers model as a way of communicating knowledge (Kasser, 2013a) pages 115-131).
- To some people, models and blueprints are those that have been used by hardware engineers in the form of schematic diagrams and sketches since the early days of engineering.
- To some people, the word model can be used to mean a reference model such as in "*We propose a [reference] model for reusability based on …*" (Prieto-Díaz, 1987).

In systems engineering, clear and concise communications is the key to success; the terminology needs to be unambiguous.

3.4. Scientific Perspective

Conclusions from the scientific perspective include:

1. MBSE conflates two distinct and different models.
2. MBSE is a poor choice of terminology for the concepts it contains.
3. MBSE suffers from a lack of holistic thinking.
4. MBSA is a return to the A paradigm or much ado about nothing new.
5. MBSE is reinventing old concepts.

Consider each of them.

3.4.1. MBSE Conflates Two Distinct and Different Models

MBSE conflates two distinct and very different information models, namely:

1. A conceptual model in the form of a vision of a FCFDS in operation. The model of the solution system under development being a significant part of the future desired situation.
2. The integrated interdependent information pertaining to both the process (project management) and product (systems and non-systems engineering) in the project that realizes the solution system which currently exists in the form of unconnected un-integrated databases and documents.

Consider each of them.

A conceptual model in the form vision of a FCDS in operation. The A paradigm in systems engineering has been using these types of models in prototype, document and simulation form for at least 50 years. Indeed, the success of systems engineering in the NASA environment in the 1960's and 1970's was attributed to a set of eight principles (Hitchins, 2007) page 85) which were updated for the 21st century (Kasser, 2013a) pages 427-437) one of which is:

"*There shall be a concept of operations (CONOPS) from start to finish of the mission describing the normal and contingency mission functions as well as the normal and contingency support functions performed by the solution system that remedies the problem*".

[3] Which is a typical characteristic of the early years of a discipline.

Now that second generation CESE tools provide ways to create a CONOPS in the form of interactive simulations and executable models, MBSE seems to be restating that capability using application language as in the definitions by Friedenthal et al. (Friedenthal, et al., 2007) and INCOSE's Vision 2020 (INCOSE, 2007) cited in Section 3.1.

When the benefits of MBSE are summarised as in the following extract (Long, 2013), the author is really summarizing the benefits of the A paradigm.

- *"Early identification of requirements issues.*
- *Missing requirements, conflicting requirements, and general defects.*
- *Enhanced stakeholder communication to enable better validation.*
- *'We fail more often because we solve the wrong problem than because we get the wrong solution to the right problem' (Ackoff) (Ackoff, 1974).*
- *Disciplined (and defensible) basis for decision making.*
- *Moving beyond "a miracle occurs here" analysis.*
- *Enhanced visibility into information gaps and system design integrity.*
- *Model-driven consistency vs. document-driven hope.*
- *Improved specification of allocated requirements to Hardware/Software.*
- *Reduction in errors reaching integration and test.*
- *Rigorous traceability from need through solution.*
- *Improved alignment of collective team understanding.*
- *One high-visibility version of truth.*
- *Reduction of rework.*
- *Improved communication & insight.*
- *Improved impact analysis of requirements changes.*
- *Knowing when you are done!"*

The integrated interdependent information pertaining to a project. The concept in the MBSE information model is to replace the 20th century independent document-centric paradigm by a paradigm in which information is stored electronically in interdependent databases where documents are views or printouts of the contents of the databases. This is a desired characteristic of the DODAF (DoDAF, 2004) as but one example. A better term for this 'model' might be an 'integrated information environment' (IIE) for the repository of project information (Cook, et al., 2001);(Kasser, 2013a) pages 97 - 104) as discussed below in Section 3.4.5.

3.4.2. MBSE Is a Poor Choice of Terminology for the Concepts it Contains

MBSE is more than just developing and using models, for example, it is also an effort to address the following process issues (Shoshani, 2010):

- ***Communication and understandability*** – well-structured models improve the ability to convey meaning to different stakeholders.
- ***Traceability*** – linked and repository based models allow for better traceability and consistent models.
- ***Early knowledge*** – early executable models allows eliciting knowledge earlier in the SDP.
- ***Reduced time to market (TTM)*** – model based analysis and design takes less time than the textual process. Models that create software deliveries automatically, improve TTM.
- ***Reuse*** – well-structured model parts can be reused in product lines or component based development, again shortening the development cycle and cost.
- ***Formal proofs*** – models can be validated early and fully, models that are turned into software code are considered proven by construct. This is useful where system high reliability is required.
- ***Maintenance*** – a model of the system captures all the data needed for change thus allowing for easier maintenance.

The term MBSE is in application language. Application language focuses on the instance described by

the application such as stating the need for a 'car' where the need is for the 'transportation' function[4]. The use of application language is limiting as well as being open to multiple interpretations as discussed above.

3.4.3.MBSE Suffers From a Lack of Holistic Thinking

MBSE suffers from a lack of holistic thinking in the following ways:

- MBSE is the future of systems engineering.
- The focus on a single modelling language.
- The focus on requirements.
- MBSE is constrained by the current paradigm.

MBSE is the future of systems engineering. MBSE proponents claim that MBSE is the future of systems engineering. However, perceptions from the Continuum HTP show that there is more than one way to perform a function, so from the Generic HTP, personnel who claim that MBSE is the way or is the future of systems engineering are just the latest example of non-holistic thinking personnel who claim their tool will fit all current and future situations. This behaviour matches Maslow's observation of non-systems thinking human behaviour which was "*I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail*" (Maslow, 1966) pages 15 and 16).

The focus on a single modelling language. Holistic thinking and its application to systems engineering is about choices.

- Perceptions from the Operational HTP suggest that the stakeholders should see the information in the model in a way that makes sense to them, rather than have to learn a modelling language.
- Perceptions from the Structural HTP point out that since UML (Unified Modeling Language™), was designed to be extendable, there was no need to develop SysML (Systems Modeling Language). All that was needed to be developed were extensions to UML.
- Perceptions from the Continuum HTP show that:
  - There are other modeling languages and other ways of performing the same function such as Object-Process Methodology (OPM) (Grobshtein and Dori, 2010).
  - There are ways to communicate the FCFDS using videos, pictures and other non-language methods as demonstrated in the Operations Concept Harbinger (OCH) which may be thought of as a multimedia combined CONOPS and system requirements repository that also contains measures of effectiveness (MOE) for each operational scenario which was developed, prototyped and demonstrated for a Force Level Systems Engineering (FLSE) application in the SEEC at UniSA (Kasser, et al., 2002).
  - While it generally is possible to use a single language to program all applications all tasks, different languages are better suited for specific applications. For example, in the early days of computing FORTRAN (FORmula TRANslation) was a language used for programming mathematical tasks, while COBOL (Common Business Oriented Language) was used to program business tasks.
- Perceptions from the Temporal HTP indicate that new languages and methods will arise in the course of time.
- Perceptions from the Generic HTP provide the lesson not learned that attempts to standardize on a single computer language have failed in the past, ADA being but one example.

The focus on requirements. Requirements are a means, not an end. The focus on requirements as an end rather than as a means stems from the roots of MBSE in the B paradigm. If, for example, the performance of a system can be documented in an executable model of a CONOPS as in the A paradigm, there is no need for most of the requirements that exist in the B paradigm.

Perceptions from the Functional perspective of the A paradigm version of the SDP show that the information in the CONOPS model is translated to requirements which are then passed on to the hard-

[4] Another example of application language is the use of 'Network centric' when the function is information distribution and the application is via networks.

ware and software design teams. The software design teams produce use cases based on the requirements. Since the CONOPS model contains the use cases, there is no need develop functional and performance requirements; the functions can be tagged with performance properties (Kasser, 2013a) pages 219 - 220). Just allow the designers to access the tagged CONOPS model and eliminate one of the major contributors to project failure, namely poor requirements.

MBSE is constrained by the current paradigm. MBSE seems to be focused on improving the 20th century process focused B paradigm rather than applying the technology to upgrade systems engineering to an improved 21st century A paradigm.

3.4.4.MBSE is a Return to the A Paradigm

Perceptions from the Generic perspective include:

1. MBSE is reinventing concepts.
2. MBSE is a poor choice of terminology for the concepts it contains.

Consider each of these findings.

3.4.5.MBSE is Reinventing Concepts

MBSE is reinventing concepts which may be new to the process focused practitioners of the systems engineering B paradigm but are well-known, and have been in use, outside their box. This situation is an example of the inadvertent use of a flawed approach to problem solving in which the research activities shown in Figure 1 have been omitted resulting in a flawed hypothesis or solution. This lack of the research step often results in the 'not Invented Here' (NIH) syndrome. The flawed process is often inadvertent because the experts in one domain are used to producing solutions without performing the research step. Consequently, when faced with a problem for which they have no immediate solution, they forget to pose the question "who has faced a similar problem?" and perform the research step to see if indeed anyone else has faced and remedied a similar problem. Thus, MBSE is rediscovering concepts that have been explored and published in prior years both within and outside of systems engineering. These concepts include:

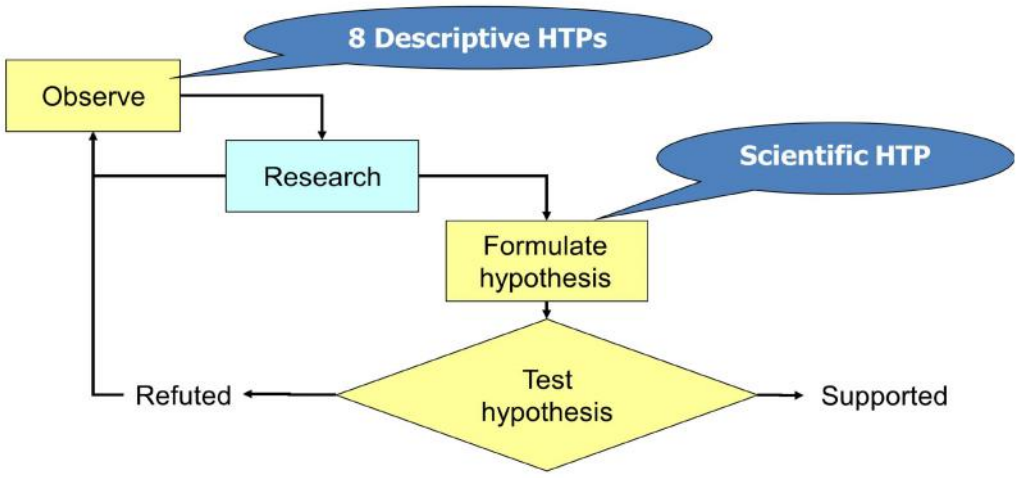

Figure 1. Systems approach to problem solving (Kasser, 2013b)

- Operations research.
- The use of models and simulations.
- Use of interdependent databases rather than in independent documents.
- The concept of an electronic executable model.

***"Operations research** is a scientific method of providing executive departments with a quantitative basis for decisions regarding the operations under their control*" (Morse and Kimball, 1951) or, "*operations research is the application of scientific and especially mathematical methods to the study and*

analysis of problems involving complex systems" (Webster, 2013). Operations research is most often used to analyze complex real-world systems, typically with the goal of improving or optimizing performance. Compare this definition with the INCOSE definition of MBSE cited in Section 3.1 (INCOSE, 2007). This overlap between operations research and systems engineering was noted as early as 1954 when Johnson wrote "*Operations research is concerned with the heart of this control problem – how to make sure that the whole systems works with maximum effectiveness and least cost*" (Johnson, 1954) page xi) a goal that many modern systems engineers would apply to systems engineering. Goode and Machol wrote that the steps of the operations research and systems engineering processes have much in common however there is a fundamental difference in approach namely "*the operations analyst is engineer is primarily interested in making equipment changes*" (Goode and Machol, 1959). A lasting difference was noted by Roy as "*Operations research is more likely to be concerned with systems in being than with operations in prospect*" (Roy, 1960) page 22).

The use of models and simulations pertaining to the FCFDS is an old concept. For example:

- Arthur D. Hall discusses the use of models and simulations (Hall, 1962) page 131).
- Modelling has long been used in the form of schematics, prototypes and scale models. For example, *"during the 1950s and 1960s electronic and hybrid analogue computers were at the heart of modelling such technological systems as aerospace and industrial plant control"* (Bissell, 2004).
- The CONOPS has been a part of systems engineering since the early days of the A paradigm.
- The 'to be' model in Business Process Reengineering.
- The conceptual model in Checkland's Soft Systems Methodology (Checkland and Scholes, 1990).

Use of interdependent databases rather than in independent documents. The concept seems to be an IIE with a number of independent and interdependent software agents, where each agent acts on the same underlying data at different stages in the SDP. Kasser and Cook discussed a proposed architecture for a frame-based third generation requirements tool (FBRET) embodying a superset of current MBSE concepts (Cook, et al., 2001);(Kasser, 2013a) pages 97 - 104) including automated approaches to requirements engineering providing features such as reasoning, structured knowledge capture, generic design solutions and elicitation assistance which were then, and still are, becoming increasingly important as the systems and the environments that they interact with, become increasingly complex.

The concept of an electronic executable model is nothing new. As but one example, a prototype of an OCH was developed, prototyped and demonstrated by the SEEC at the UniSA (Kasser, et al., 2002). In early 2002, the Joint Systems Branch (JSB) of the Australian Defence Science and Technology Organisation (DSTO) was carrying out a number of initiatives in architecture and systems engineering for joint C4ISREW systems analysis, future capability studies and force development. The teams working in these initiatives had produced many interesting and important findings, useful data and references for supporting force level defence capability planning and management, and had gained good experience in using various tools for architecture development and analysis. JSB was then faced with the problem of how to bring these outcomes together and indicate the feasibility of FLSE across the strategic and capability disciplines in a relatively simple demonstration. The dimensions of the problem included:

- Process and process interactions.
- Tools required at each level.
- Communications and information flows.
- Data, change and configuration management.
- Organizational cultural imperatives.

The solution developed at SEEC was to present the outcomes in the form of a prototype *F*orce-*L*evel *A*ustralian Defence Force *S*ystems *H*arbinger (FLASH) based on the OCH concept. The FLASH was not based on any specific language; rather it was based on the concept of storing information in an underlying database and providing stakeholders with various views in their own languages.

In conclusion MBSE seems to be much ado about nothing new!

4. Back to the Future?

MBSE needs to apply some critical thinking and reflect on itself as well as applying some holistic thinking to model itself. Even if it does not change its focus, at the least it needs to add some intelligence into its products. It can do this by applying the findings from research that has already been performed and published as well as performing new research. This section summarizes some of the research performed at SEEC in the UniSA that together with suitable findings from other research programs could and should be applied in the development of third and fourth generation CESE tools with smart capabilities. They include:

1. Distinguishing between operational and functional models and simulations
2. Applying the concept of inheritance.
3. Integrated project databases.
4. Adding intelligence to the tools.

4.1. Distinguishing Between Operational and Functional Models and Simulations

There is a need to distinguish between operational and functional models and simulations. Reliance on functional models and simulations to save the costs of testing and evaluation can be dangerous since there is a major difference between operational and functional models and simulations.

Operational models and simulations focus on actual and conceptual views of the '*what*', where:

- Actual views include models of *what* the system is doing.
- Conceptual models include models of *what* the system can do, *what* the system should do and *what* the system needs to do. These models can be used to gain consensus on the '*what*' aspect of a system.

Functional models and simulations focus on '*how*' it is being done. These are useful when the underlying mechanisms are well-understood and the functionality can be expressed mathematically. However, when the underlying mechanisms in an unprecedented system such as a new aircraft are unknown, then using simulations as training tools can be downright dangerous. The model and simulation is only as good as its underlying assumptions, and when they are wrong, people can be killed.

4.2. Applying the Concept of Inheritance

Inheritance may be used in ways that extend the software engineering usage. Traceability is an inheritance function. Design elements may be traced back to scenarios in the CONOPS, requirements, regulations, etc. Inheritance is a major advantage of the object-oriented systems engineering paradigm since from the Generic perspective most new systems are similar to, or can be considered as a class of, an existing system. For example, a communications satellite is a type of spacecraft, a destroyer is a type of surface ship, and a new car is similar (if not almost identical) to the previous model. Requirements reuse is becoming desirable. However, reuse is carried out in an ad-hoc manner (Von_Knethen, et al., 2002)[5]; the person in charge copies an old document and edits or enhances all parts they consider relevant. They integrate parts from other documents that deal with functionalities they have to add. Now, inheritance could be implemented as:

- A function that identifies the type (class) of system and inherits requirements from a standard database for that type (class) of system. This functionality would maximize the completeness of the requirements by ensuring that applicable non-system specific performance requirements are not overlooked[6].
- A function that allows selected requirements to be copied from one database to another as was demonstrated in TIGER (Kasser, 2013a) page 231).
- Templates containing selected information for specific types of document printouts from the database.

[5] How much has this changed in the last 12 years?

[6] In the A paradigm, these would be the non-functional and non-performance system requirements such as environmental, vibration environmental and other regulatory requirements.

- Standard templates for specific types of systems (Kasser, 2013b) pages 191 - 196).

4.3. Integrated Project Databases

Researchers at SEEC at UniSA researched the B paradigm SDP from an information perspective to try and integrate the information associated with systems engineering and project management. The SDP was considered as three interdependent streams of work between milestones as shown in Figure 2 (Kasser, 1995) page 140). The product and process information associated with the streams of work as being necessary for effective system and software development based on/derived from the requirements was summarised in the form of a set of Quality System Elements (QSE) (Kasser, 2013a) 97 - 104) which include:

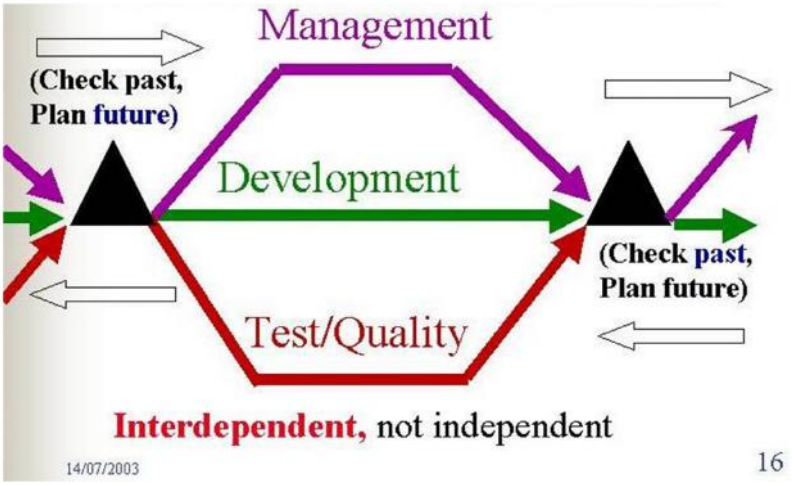

Figure 2. The Three streams of work

- ***The Requirement statement*** - the imperative statement containing both the required functionality and its corresponding Quality criteria or other form of representation.
- ***Unique identification number*** - the key to tracking.
- ***Traceability to source(s)*** - the previous level in the production sequence.
- ***Acceptance criteria*** – the answer to the question "how will we know the requirement has been met?"
- ***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.
- ***Risk*** - any process and product risk factors associated with the requirement.
- ***Traceability to implementation*** - the next level in the production sequence. Thus in software for example, requirements are linked to design elements, which are linked to code elements, and so on.
- ***Estimated cost and schedule*** - these feed into the management plan and are refined as the project passes through the SDP.
- ***The level of confidence in the cost and schedule estimates*** - these should improve as the project passes through the SDP.
- ***Rationale for requirement*** - the extrinsic information and other reasons for the requirement.
- ***Planned verification methodology(s)*** - developing this at the same time as the acceptance criteria avoids accepting requirements that are either impossible to meet or too expensive to verify.
- ***Keywords*** - allow for searches through the database when assessing the impact of changes.
- ***Production parameters*** - the Work Package (WP) elements in the Builds in which the requirements are scheduled to be implemented.
- ***Testing parameters*** - the Test Plans and Procedures in which compliance to the requirements are scheduled to be verified.
- ***Traceability sideways to document duplicate links*** - required when applying the QSE to an exist-

ing documentation-based project.

- ***Access control parameters*** – national security classification or company confidential as appropriate.
- ***Version control*** – for use by configuration management in tracking changes. This is the version number, copies of older versions of the requirement and links to the CCB change database.
- ***Non-functional elements of capability needed*** – survivability, reliability, maintainability, etc. tagged to the scenarios in the CONOPS
- ***Risk mitigation*** – the strategies for mitigating the identified risks. This property has links to the WPs.

From an object-oriented perspective, the QSE can be considered as properties of requirements since requirements drive the realization process. A WP template for each element in the realization process is shown in Table 1 (Kasser, 2010). The WP provides for linkage of all work or activities in the realization process to the requirements and facilitates identifying missing activities and products.

Table 1 Work Package (WP) content template

1	Identification number
2	Name of activity
3	Priority (based on priority of requirement)
4	Narrative of activity
5	Schedule (± accuracy)
6	Products (outputs)
7	Acceptance criteria for products
8	Estimated cost (± accuracy)
9	Reason activity is being done
10	Prerequisites (inputs - prior products or milestones)
11	Resources (people, equipment, material)
12	Internal key milestones (if any)
13	Decision points (if any)
14	Process and product risks (probability, seriousness, and mitigation WP ID)
15	Requirements Traceability (the requirement this WP is realizing)
16	Lower level work package ID's (if any)
17	Assumptions not stated elsewhere

4.4. Adding Intelligence to the Tools

The object-oriented paradigm encapsulates processes (functionality) as well as data into an object. MBSE needs to adopt the concept. Expert system technology could be used to build human experience and tacit knowledge into both the process and product part of the object. This could add intelligence to systems engineering and project management tools. For example, consider the following types of intelligent functions that might be encapsulated within third generation CASE and requirements management tools.

1. Text clarification of some types of poorly written requirements.
2. Feasibility checker.
3. Property correlation.
4. Risk reduction and monitoring.
5. Property completeness.
6. Progress monitoring.
7. Facilitating and ensuring the completeness of testing some types of poorly written requirements.

Text clarification. This function could scan the wording of the requirement and flag any requirements that are poorly written. Research at SEEC developed a prototype software tool that performed this

process (Kasser, 2002). An updated tool to ingest and elucidate requirements (TIGER)[7] has been used in class lectures on requirements engineering in several postgraduate courses. Before TIGER was introduced, the discussions in the tutorials focused 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).

Feasibility checker. This function could check the feasibility of the requirements from a testing perspective and flag those that were not feasible at the time they were written before they were accepted.

Property correlation. This function could correlate properties of requirements in the QSE and provide an indication when something needs further examination. For example, the function could correlate:

- ***Estimated cost to implement with priority***. The customer could be asked if a requirement with high estimated cost to implement and low priority is really needed.
- ***Risk with priority.*** The customer could be asked if a requirement with high implementation risk and low priority is really needed.
- ***Text of requirement with acceptance criteria*** and identify requirements without corresponding acceptance criteria.

Risk reduction and monitoring. This function could ensure that all risks have mitigating strategies and each strategy is implemented in a WP element, and provide appropriate warnings for those that are incomplete.

Property completeness. This function could check for the presence or absence of other properties and provide indicators. For example, it could check that all requirements have acceptance criteria, priority, risk and cost estimate properties.

Progress monitoring. The question "*what do you mean, you can't tell me how much of my project has been completed?*" is a very difficult one to answer in the current paradigm (Kasser, 2013a) pages 65-74). However, the use of Categorized Requirements in Process (CRIP) charts can provide a better answer to the question than the measurements made in the current paradigm. They can also provide early identification of anomalies in the implementation process.

Facilitating and ensuring the completeness of testing some poorly written requirements. This function could automate a manual process of building test compliance matrices. It would convert written requirement paragraphs containing multiple requirements into separate requirement paragraphs. As an example of the work that this tool could expedite, consider the following requirement (ST-DADS, 1992):

```
204.1 DADS shall automatically maintain statistics concerning the number of
times and the most recent time that each data set has been accessed. These same
statistics shall be maintained for each piece of media in the DADS archive.
```

The function could split this requirement into the following four requirements to simplify tracking the completeness of the test plans:

```
204.1a   DADS shall automatically maintain statistics concerning the
         number of times ~~and the most recent time~~ that each data set has
         been accessed. ~~These same statistics shall be maintained for
         each piece of media in the DADS archive~~.
204.1b   DADS shall automatically maintain statistics concerning ~~the
         number of times and~~ the most recent time that each data set has
         been accessed. ~~These same statistics shall be maintained for
         each piece of media in the DADS archive.~~
204.1c   DADS shall automatically maintain statistics concerning the
         number of times ~~and the most recent time~~ that ~~each data set has
         been accessed. These same statistics shall be maintained for~~
         each piece of media in the DADS archive [has been accessed].
204.1d   DADS shall automatically maintain statistics concerning ~~the~~
```

[7] Available at http://therightrequirement.com.

~~number of times and~~ the most recent time that ~~each data set has been accessed. These same statistics shall be maintained for~~ each piece of media in the DADS archive *[has been accessed].*

Leaving the sections of the requirement that were not being tested in place but stricken through would clearly identify which section of the requirement is being tested. An unfortunate side effect is that it would also clearly show the defects in the requirement. Note that the phrase *'[has been accessed]'* has been moved in the last two sub-requirements to clarify the sub-requirement.

5. Summary

This paper perceived MBSE from some of the HTPs on the perspectives perimeter and presented the findings from the research. The paper showed that:

- MBSE is displaying some of the characteristics of an emerging disciple.
- While MBSE is a laudable attempt by systems engineers in the B paradigm to return to the original A paradigm, MBSE lacks the awareness of how current information technology could smarten up systems engineering tools.
- Past research at SEEC at UniSA is among the research that has produced functionality that could and should be introduced into next generation tools containing models and simulations.
- The focus of MBSE is inwards on:

  1. Improving the process using the concept of the IIE.
  2. Creating a model of a CONOPS using SysML.

6. Conclusions

In conclusion MBSE seems to be much ado about nothing new! MBSE needs to apply some critical thinking and reflect on itself. It needs to:

1. Apply some holistic thinking to itself starting with the Big Picture perspective of where system engineering takes place in the acquisition and development of systems.
2. Apply the Generic perspective to incorporate some 'out of the [systems engineering] box' technology and techniques from [the box] in the information technology domain to smarten up its models.

7. Author's Biography

Joseph Kasser has been a practicing systems engineer more than 40 years and an academic for about 16 years. He is a Fellow of the Institution of Engineering and Technology (IET), an INCOSE Fellow, the author of "*Holistic thinking: creating innovative solutions to complex problems*", "*A Framework for Understanding Systems Engineering*" and "*Applying Total Quality Management to Systems Engineering"* and many INCOSE symposia papers. He has received a number of awards for performing and directing systems engineering including NASA's Manned Space Flight Awareness Award (Silver Snoopy). He holds a Doctor of Science in Engineering Management from The George Washington University. He is a Certified Manager and holds a Certified Membership of the Association for Learning Technology. He also started and served as the inaugural president of INCOSE Australia and served as a Region VI Representative to the INCOSE Member Board. He has performed and directed systems engineering in the UK, USA, Israel and Australia. He gave up his positions as a Deputy Director and DSTO Associate Research Professor at the SEEC at the University of South Australia in early 2007 to move to the UK to develop the world's first immersion course in systems engineering as a Leverhulme Visiting Professor at Cranfield University. He is currently a Visiting Associate Professor at the National University of Singapore.

8. References

Ackoff, R. L., Redesigning the Future: A Systems Approach to Societal Problems, John Wiley & Sons, New York, 1974.

Bissell, C., 2004, A great disappearing act: the electronic analogue computer, proceedings of IEEE Conference on the History of Electronics.

Checkland, P. and Scholes, J., Soft Systems Methodology in Action, John Wiley & Sons, 1990.

Cook, S. C., Kasser, J. E. and Asenstorfer, J., 2001, A Frame-Based Approach to Requirements Engineering, proceedings of 11th International Symposium of the INCOSE.
Daniels, J., Bahill, T. and Botta, R., "A Hybrid Requirements Capture Process," the 15th annual International Symposium of the INCOSE, Rochester, NY., 2005.
Deming, W. E., The New Economics for Industry, Government, Education, MIT Center for Advanced Engineering Study, 1993.
DoDAF, DoD Architecture Framework Version 1.0, 9 February 2004, 2004.
Friedenthal, S., Griego, R. and Sampson, M., 2007, INCOSE Model Based Systems Engineering (MBSE) Initiative, proceedings of the 17th International Symposium of the INCOSE.
Frittman, J. and Edson, R., 2010, Illustrating the Concept of Operations (CONOPs) Continuum and its Relationship to the Acquisition Lifecycle, proceedings of Naval Post Graduate School. 6th Annual Acquisition Research Symposium.
Goode, H. H. and Machol, R. E., Systems Engineering, McGraw-Hill, 1959.
Grobshtein, Y. and Dori, D., "Generating SysML Views from an OPM Model: Design and Evaluation " the 19th Annual International Symposium of the International Council of Systems Engineering Chicago, IL., 2010.
Guo, J., "Incorporating Multidisciplinary Design Optimization into Spacecraft Systems Engineering," 8th Annual Conference on Systems Engineering Research, Hoboken, NJ., 2010.
Hall, A. D., A Methodology for Systems Engineering, D. Van Nostrand Company Inc., Princeton, NJ, 1962.
Hitchins, D. K., Systems Engineering. A 21st Century Systems Methodology, John Wiley & Sons Ltd., Chichester, England, 2007.
INCOSE, Systems Engineering Vision 2020, INCOSE-TP-2004-004-02 (Version 2.03), INCOSE, 2007.
Jenkins, G. M., "The Systems Approach," Systems Behaviour, J. Beishon and G. Peters (Editors), Harper and Row, London, 1969, p. 82.
Johnson, E. A., "The Executive, the Organisation and Operations Research," Operations Research for Management, Volume 1., J. F. McCloskey and F. N. Trefethen (Editors), The Johns Hopkins Press, Baltimore, 1954.
Kasser, J. E., Applying Total Quality Management to Systems Engineering, Artech House, Boston, 1995.
---, 2002, A Prototype Tool for Improving the Wording of Requirements, proceedings of the 12th International Symposium of the INCOSE.
---, "SDM5004 Systems Approach to Project Management," National University of Singapore, Singapore, 2010.
---, A Framework for Understanding Systems Engineering (2nd Edition), CreateSpace Ltd., 2013a.
---, Holistic Thinking: creating innovative solutions to complex problems, Createspace Ltd., 2013b.
Kasser, J. E., Cook, S. C., Scott, W., Clothier, J. and Chen, P., 2002, Introducing a Next Generation Computer Enhanced Systems Engineering Tool: The Operations Concept Harbinger, proceedings of SETE 2002.
Kasser, J. E., Tran, X.-L. and Matisons, S., 2003, Prototype Educational Tools for Systems and Software (PETS) Engineering, proceedings of Proceedings of the AAEE Conference.
Long, D., "Faster, Better, Cheaper – The Fallacy of MBSE?," the 2013 Systems Engineering and Test and Evaluation Conference (SETE 2013), Canberra, Australia, 2013.
Long, D. and Scott, Z., A Primer for Model-Based Systems Engineering, Vitech Corporation, 2011.
Maslow, A. H., The Psychology of Science, Harper and Row, 1966.
MBSE, Model-Based Systems Engineering FAQ, The Model-Based Systems Engineering Forum, 2011, http://www.modelbasedsystemsengineering.com/mbse-faq/, accessed on 18 July 2013.
Morse, P. M. and Kimball, G. E., Methods of Operations Research, The Technology Press of Massachusetts Institute of Technology and John Wiley & Sons, New York, 1951.
Nelson, G., 2007, The ConOps in a Self-Similar Scale Hierarchy for Systems Engineering, proceedings of Conference on Systems Engineering Research.
Prieto-Díaz, R. (Editor), Classification of Reusable Models published in Software Reusability, 1987, ACM Press, 1987.

Roberts, N., "An Analysis of Concept of Operation Development," Steven's Institute of Technology, 2008.

Roy, R. H., "The Development and Future of Operations Research and Systems Engineering," Operations Research and Systems Engineering, C. D. Flagle, W. H. Huggins and R. H. Roy (Editors), Johns Hopkins Press, Baltimore, 1960.

Shoshani, S., Effective use of MBSE, Israel Institute of Technology, 2010.

ST-DADS, ST-DADS Requirements Analysis Document (FAC STR-22), Rev. C, August 1992, as modified by the following CCR's:- 139, 146, 147C, 150 and 151B, NASA/Ford AeroSpace, Greenbelt, MD, 1992.

Sutcliffe, A., Galliers, J. and Minocha, S., 1999, Human Errors and System Requirements, proceedings of IEEE International Symposium on Requirements Engineering.

Von_Knethen, A., Paech, B., Kiedaisch, F. and Houdek, F., 2002, Systematic Requirements Recycling through Abstraction and Traceability, proceedings of IEEE Joint International Conference on Requirements Engineering.

Webster, Merriam-Webster Online Dictionary, 2013, http://www.merriam-webster.com/dictionary/operations%20research, accessed on 6 August 2013.