Document the Model, Don't Model the Document

David Harvey
Aerospace Concepts Pty Ltd
david.harvey@concepts.aero

Michael Waite
Aerospace Concepts Pty Ltd
michael.waite@concepts.aero

Paul Logan
Empel Solutions Pty Ltd
pwlogan@ozemail.com.au

Tommie Liddy
Aerospace Concepts Pty Ltd
tommie.liddy@concepts.aero

Abstract. The document is a universally accepted method for conveying information when defining complex systems. It allows a wide range of stakeholders to comprehend the areas that pertain to their interests; however, a document lacks explicit traceability. To capture and maintain that information there has been a consistent rise in the use of computer models to support the definition and design of complex systems. While being useful to clearly define and manage interactions of the stakeholders, system and environment within the systems engineering team, the model is often a mystery to those not involved in its creation. This lack of clarity has led to an ideological battle between document-based and model-based systems engineering, with the former group claiming better understanding and communication and the latter claiming clearer, consistent and traceable definition. However, all systems engineering is model based, with the model either in the systems engineer's head or in a software tool. This 'basis' argument does not address that which is important: whether the focus is on the document (document-centric) or on the model (model-centric). This paper proposes that there is a continuum of centricity when using a model-based systems engineering tool; from entirely document-centric where the tool is used to model the systems engineering documents required, through to entirely model-centric, where the tool is used to make a model of and about the system, then this model is reported upon to create the document. A basic example describes two points on this continuum. The authors further propose that a model-based, model-centric approach provides a range of benefits; the systems engineering effort can go towards defining the system, gaining the benefits of rigour, traceability and consistency, and that views of this information can be produced, often as documents, allowing clear communication with a range of stakeholders.

INTRODUCTION

Systems engineering provides a structured approach to definition, design and implementation of systems that address defined user problems. The nature of systems engineering means this analysis and design is built upon a holistic consideration of the entire system, together with interacting external systems and the environment in which they exist across the complete lifecycle of the system (ISO 2008). The origins of the discipline can be traced back to major projects in the 1930s, with the holistic approach to complicated systems such as air defence networks (Haskins 2011). The tools used by these early systems engineers were the classic pen and paper. As the complexity of the systems and design teams responsible has increased, there has been a move away from simple pen and paper support through word processors to computer based systems engineering tools. The use of these tools does not take the place of experienced systems engineers, but supports their efforts to define user needs, system requirements, design and implementation aspects of the system. This 'outsourcing' of the recording of systems knowledge, while retaining the understanding of how to find the information and how it is structured is analogous to a wider move to the outsourcing of searchable information, with humans having to remember less, 'know' less, but being able to access much more information using Google (Sparrow *et al.* 2011). These MBSE tools are being used in a variety of

ways – from simple replacement of pen and paper through to full utilisation of the unique functionality they provide. This paper discusses the continuum of MBSE tool usage from solely recording progress on a required document set, through to developing a model of and about the system with any required documents, reports and other information artefacts produced as views on the model. Examples are presented to illustrate these diametrically opposed paradigms. The authors further propose that when implemented fully, a model-centric method provides the engineering benefits of model-based systems engineering while not neglecting, and in fact enhancing the communication benefits of traditional documents.

MODEL BASED SYSTEMS ENGINEERING

'Model-based' systems engineering methodologies are not new, and it can be argued that the field of Systems Engineering is inherently model-based in nature (Crisp 2007). From its earliest beginnings, sound systems engineering has had a model at the centre of system development – this holistic approach underpins the definition of systems engineering as a field. In the early days of systems engineering, this model was contained in the systems engineer's head. However, as project complexity increases, the maximum number of relationships between information increases as a square of the number of elements, and as such, the complexity of the analysis increases. Metcalfe's law shows that a fully-connected mesh with ***n*** nodes that have a single relationship with all other nodes will have ***n(n-1)/2*** relationships (Metcalf 1973). In a requirements specification that has 1000 requirements (not uncommon for a large project), the number of possible connections between these requirements is close to half a million. While a system representation with a well-structured set of requirements, being in essence hierarchical rather than meshed may reach only a fraction of this degree of complexity, it nonetheless very difficult for an individual systems engineer to hold the model in their head.

There is another complicating factor; large scale projects are usually developed by teams of systems engineers. As can be seen from the calculation above, this difficulty with sheer number of relationships can be true even for smaller sections of a project, such as sub-systems of the system of interest. Not only does each engineer need to hold a detailed representation of their sub-system of interest, but they also need at least a basic representation of the entire system to keep track of interactions. This is essential, as the interaction of related subsystems and environments is one of the main determinant characteristics of system complexity and the *raison d'être* of systems engineering. Given that human cognition has limits – Miller proposed that the 'magical number' that limits human cognitive ability to process information is seven, plus or minus two at one time (Miller 1955) – the ability of humans to keep track of the complexity and sheer number of interactions in the definition of a modern system is questionable.

Faced with the need to deal with complexity of system and interaction definition and with having to work in teams to define systems, engineers have sought tools to assist in system development. These tools assist in keeping track of the problem and solution as defined, as well as interactions between the various information elements. Originally these tools were purely pen and paper, used to support the engineer's understanding of the problem and system solution. The level of support progressed through improved technology and the application of word processors, which allowed the construction of a series of documents to define a system and its environment. These tools, which have some simple linking capability (e.g. the ability to define a link – a 'cross-reference' – to a figure caption or a reference document), were used to "computerise" the process which was previously done with pen and paper. This approach also supported the standard committee-based acquisition process for large projects. This constraint is inherent with the nature of large projects and the understandable interest of company management and government oversight where appropriate. A further step in the tool support of systems engineering was the implementation of computer based systems engineering tools (White *et al.* 1991). These tools have progressed from these beginnings to develop into the Model Based Systems Engineering (MBSE) tools of today. The systems engineer can use these tools to focus on defining a model of and about a system, rather than on populating any required definition documents. This paper postulates that there are many benefits to be gained by taking a model-centric approach and appropriately applying MBSE tools to exploit their extended capabilities, rather than

simply replicating the document-focussed approach from the past in a new tool.

MODEL-CENTRIC VS DOCUMENT-CENTRIC

Model-Based Systems Engineering tools are now widely used. Much of the systems engineering community is increasingly using information technology support to deal with the complexity inherent not only in system definition but also in the more interactive, teaming nature of their projects. However, there is another aspect to consider – using a MBSE approach **does not** necessarily ensure that the focus of work is on **the model**. The 'centricity' of focus is a very important distinction in how MBSE methods are implemented. To complicate matters more, this is not a simple binary attribute but rather there is a continuum of centricity.

At one extreme of model centricity, the focus is entirely on the required documents. The other extreme is a focus entirely on development of information of and about the system – the model. These two extremes are shown in Figure 1 graphed against the basis of the method, whether it be model-based or document-based. As indicated in this matrix, it is possible to approach a problem in a document- or model-based manner with a document- or model-centricity. At the bottom left, the problem is addressed using a document-centric, document-based method. All definitional effort is put into completing a mandated or chosen set of systems engineering products, with no central model of the system or its definition. This is a very naïve view of systems engineering, and it can be argued that this approach does not even constitute systems engineering, as it does not consider the system and its stakeholders and environment holistically.

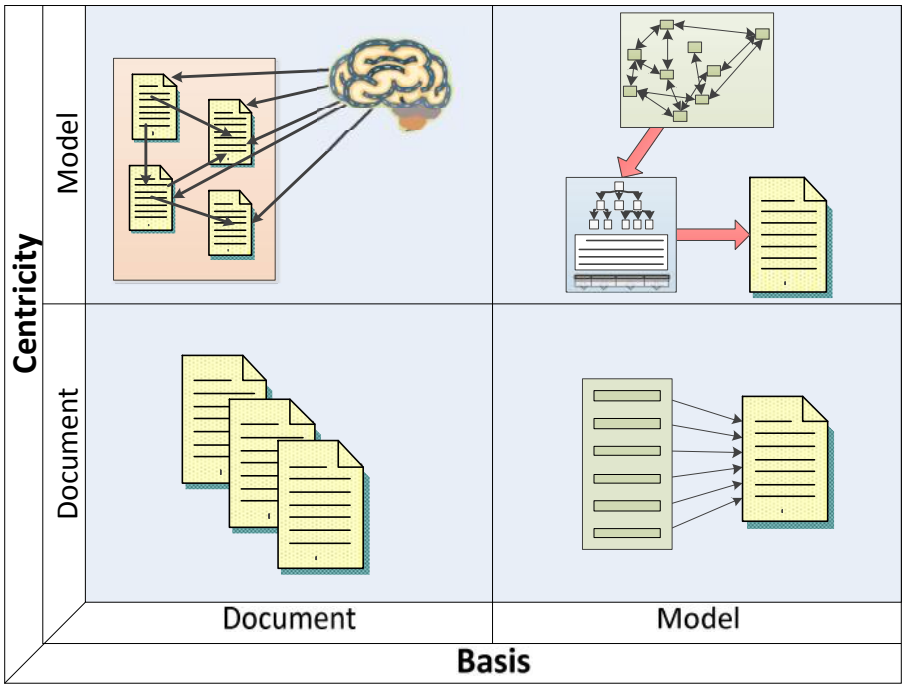

Figure 1. Systems engineering centricity versus basis

The model-centric, document-based approach, represented in the top left-hand corner of Figure 1, the only holistic model of the system is in the systems engineer's brain. This model is then used to populate the sections of the required document set. Unless the system engineer is a truly unique individual, only five to seven independent concepts are maintained simultaneously. This then is a very limited view of the system and a very dubious approach to system analysis and definition.

The extremes of the centricity continuum within the model-based approach are displayed on the right of the diagram. The bottom right represents the extreme situation where the paragraphs and sections of the required document (or documents) are modelled in the MBSE tool with no relation to each other (other than that defined by the document template or Data Item Description) and output to create the document when required. The top right represents the entirely model-centric model-based

approach where a model, which is an integration of an engineering model "about" a system as well as a technical model "of" a system, is created in a MBSE tool and querying and traversal of this model is used to create document artefacts as needed at any time in the definition process.

The centricity of approach has a significant effect on the benefits that can be gained through application of sound systems engineering. An important distinction between the two ends of the spectrum arises from the basic truth that the model is a *representation* of something, whereas the document is a *report* on something. The model-centric techniques create a model that approximates reality (the system) to varying degrees including elements, interfaces and the relationships between the elements in a centralised source of knowledge, the model. The model is the artefact produced and the focus of the systems engineering effort is to get the model to represent **The System** at an appropriate level of detail. This approach has notable positives such as enhanced traceability, consistency and the ability to be verified and validated (with the creation of an executable model). The purely document-centric approach has the focus of work entirely on the documents required to define the system, normally such that the project can pass a series of committees. This approach, even when undertaken using a MBSE tool, does not bring with it the traceability and consistency benefits of the model-centric approach. The extremes of this continuum, along with explanations and examples are discussed in the following sections.

Document-centric

Documents are a useful way to communicate with stakeholders. In fact, they are often the best and only way to communicate with stakeholders, particularly those from outside the core systems engineering team. For this reason (among others) they continue to have an important role to play in systems engineering (Logan & Harvey 2011). A document is simply an assemblage of data elements, arranged in a logical order for the convenience of the reader – or at least it should be, made fit for purpose to address the reader's concerns or interest in the system (ISO 2011).

Since antiquity, engineers have recorded their work on media (such as paper) that allow the communication of their ideas to others. "Document-centric" is the term applied to the approach used where the document author(s) directly enter the information into the end-product, whether a physical document, a word processing package or a model of a document in a MBSE tool. Using this approach, the author may write the document from start to finish, or alternatively write the sections in any order they choose, as the information becomes available. Where this newly entered information relates to other information already contained in the document, or to an external source, that relationship can be articulated through a variety of methods. Whether this includes footnotes, uniquely numbered lists, references, cross-references to tables and figures, or explicitly identifying the relationship in full, the relationship can be captured. The linkage between information, and therefore traceability to the origin of information and decisions, is only as good as the relationships that are explicitly identified. Any information that is not articulated in this way only exists in the mental model that the author has of the document. Artefacts within the document such as diagrams, as well as any other information stored, are static. This means that if related information (or indeed the same information that appears in multiple places in a document) is updated, the engineer needs to identify and find anything that is impacted throughout their document and change that too. With even a mildly complicated system, this impost can rapidly become time consuming and very much a source of error and inconsistency.

A graphical representation of a document-centric systems engineering approach is shown in Figure 2. This shows a system design process under a document-centric approach; at each stage a set of documents are produced, with no reference to a team central model. The input to each stage is then the set of documents from the stage before. They serve as record of the analysis and design, as well as driving the process. This focus on the artefact to be delivered, often mandated to get through a review and decision making process, can lead to inconsistency and incompleteness between the information, difficulty in conducting change impact assessment and performance analysis, minimal re-usability of the project information and increases the risk of projects being over-budget and late due to unexpected errors which need to be remedied (often late in project timeline). The overall objective of a document is to communicate accurate information, upon which decisions can be made.

However, when the focus of the engineer becomes the production of these documents, and moves from the definition of an appropriate system and the solution of a problem, this increases the risk of mis-steps in the design and analysis.

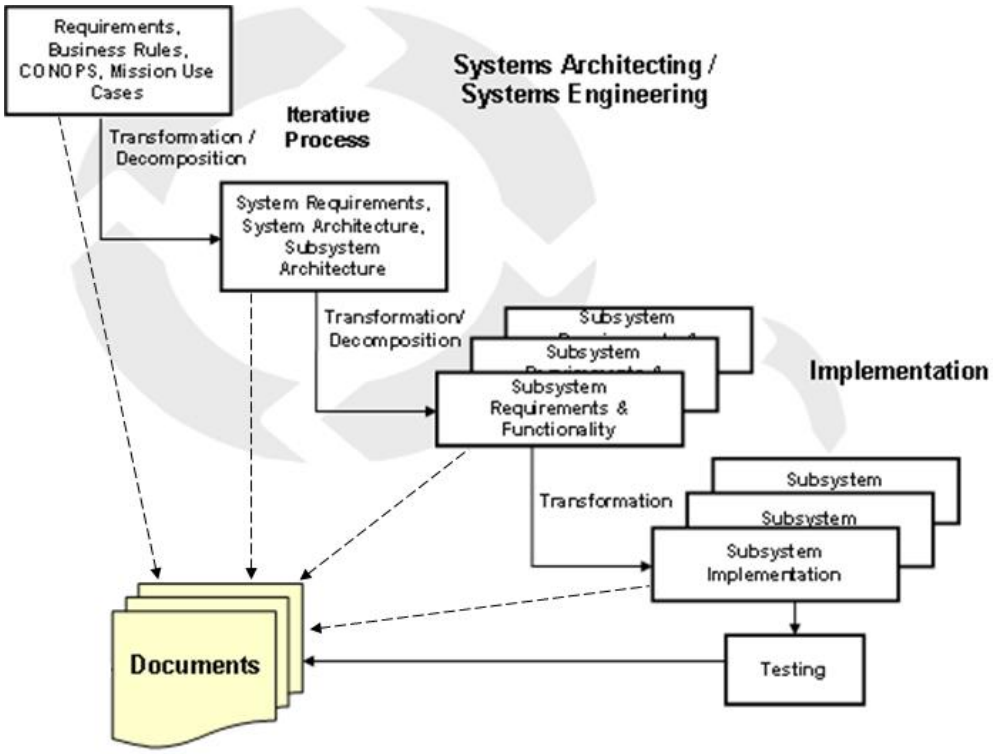

Figure 2. Systems Engineering Process – Document-centric (suggested by Estafan 2008)

Model-centric

In the model-centric approach to systems engineering, model development is the major focus of the engineer's effort. The goal is to create a model of and about a system; linking together and within a model of the problem space (Logan & Harvey 2010, Waite & Logan 2011), requirements of a systems solution and even system design. In this approach the model is the key artefact, with the production of documents becoming of secondary importance. This is not to say that documents are not used, or indeed are not important. They can be created at any time as a defined snap-shot in time of a slice of the model. The use of a series of appropriate queries on a database model that traverse the model to gather the required information contained within and output it in a way that is fit-for-purpose for review or further input. These outputs may include mandated documents (such as the Operational Concept Document in the Australian Defence environment), development templates showing all the information in a particular information class or any number of *ad hoc* documents, tables or views as required by stakeholders.

If there is a focus on the document, even if a MBSE tool is used to create a model of a document, only the information required to populate the document is captured. When the model of the system is the focus (as in the model-centric approach) the aim is to capture information that adequately represents the system and the problem which it is designed to address. This may create larger overhead, particularly for simple systems, due to the amount of information captured. However from this model of the system, various stakeholders are able to produce an array of different documents / views of this information which are necessarily consistent with one another at any one time. Focussing on the documents can produce quicker results, particularly for simpler systems, but it does not bring these very real engineering benefits, which should be the focus when system engineering is used to design complex systems which address real world problems.

A model-centric approach to developing a system is shown in Figure 3. Each of the systems engineering tasks contributes to the construction of a common model. This model is progressively defined as the analysis and design development moves towards the production of an appropriate system. This common model provides benefits of enforced traceability, consistency and clear support to verification and validation. Whenever documents are required, they can be produced as artefacts that reflect the current state of the information stored within the central model. In this way, the objective of the systems engineering work is not on creating the document but on creating a model of the system that can be reported on as required. With appropriate reporting facilities this focus on the engineering can be conveyed to the stakeholders in a form with which they can interact.

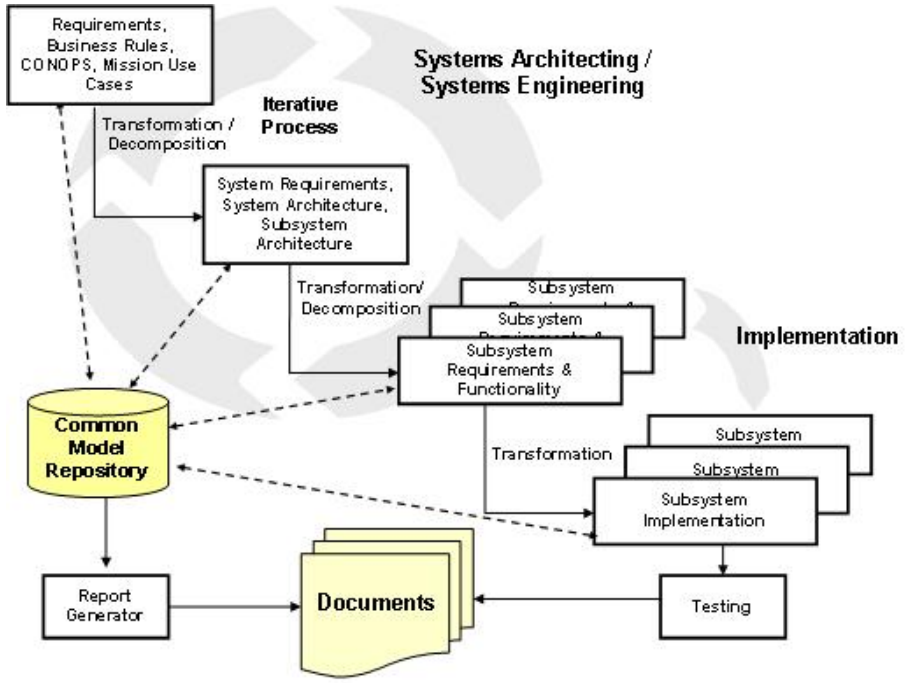

Figure 3. Systems engineering process - Model-centric (derived from Estafan 2008)

A BASIC MODEL

As an example a simplified version of a model schema segment is shown in Figure 4. It must be noted that each of the model element classes have many more relationships to other elements in the model, but for the sake of simplicity only the relationships pertinent to this example are shown.

In this schema, elements of the model are arranged in classes (elements with the same available attributes and relationships) and relationships (the defined connection between element classes). Classes in this diagram are shown as white rectangles, and relationships are shown as the arrow connecting them. These relationships are bi-directional, and therefore do not represent "flow" (like in a flow-block diagram), but show how the elements are related e.g. a "Performer" *performs* an "Activity", but also (relationship not shown) the "Activity" is *performed by* a "Performer".

By constructing the model in this fashion, data is gathered, analysed and recorded in the model as it becomes available. The traceability of the information can easily be tracked and understood by the system modeller and impact analysis can take place when changes are made. It is also easy to test the robustness of model elements by checking that the backwards traceability exists such that model elements have not been erroneously introduced. For instance, if the model contains a system component which performs a function, but that function is not based on a user need, then an issue is raised. The issue can then be resolved by the engineer – is the relationship to a user

need missing and resolved by creating that link, or is the function (and therefore component) superfluous? Of course the legitimacy of the elements and their relationships cannot be automatically tested; this relies on the provenance of the source data and the expertise of the system modeller (i.e. rubbish in, rubbish out). Figure 4 also depicts an instantiation of this structure showing the elements which model a typical restaurant scenario. In this case, a diner (the "performer") orders food (the "Activity"). In order to do so, the diner needs to understand what the different options are from the restaurant (the "Need"). To satisfy its customers, the restaurant solution concept will have to possess the ability to inform the patrons of what food is available (the "Function") which will be allocated to a menu (the "Component").

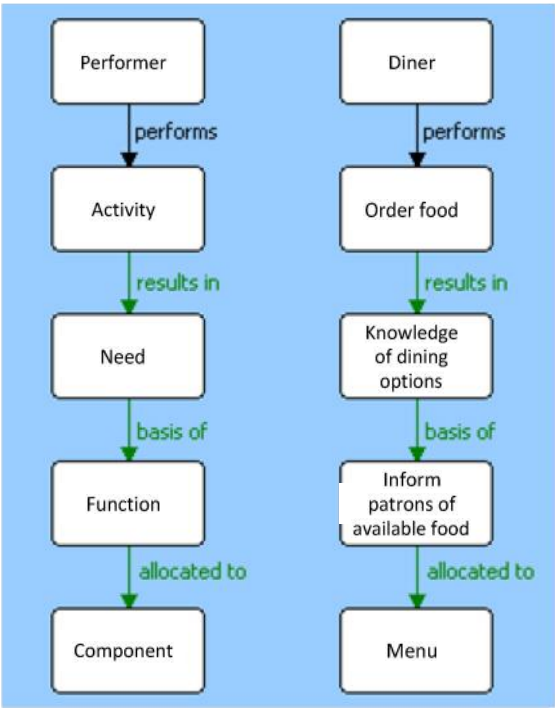

Figure 4. Structure of information

This representation is a model of the system of interest. The information contained in this model is agnostic of the documentation, diagrams and other views that will ultimately be generated from this model. The data in the system model are structured in an architecture which is organised in a logical manner. As project information is captured, relevant information is recorded within elements as their attributes, and their relationships with other elements. The structure shown in Figure 4 can readily be followed from one element to the next clearly showing how each data item relates to one another. However, when the information is generated into a document, the information is reported in the order specified by that document's format.

A COMPLEX DOCUMENT

An example of document generation as a database report is shown in Figure 5, which depicts the relationship between a simplified model structure and the structure of the Australian Defence Operational Concept Document (OCD) version 1.4 (Department of Defence, 2009). The structure of the OCD has been revised and the Data Item Description has been reissued as version 2.0 dated 30 November 2011 (Department of Defence, 2011). Often document templates require the same information to be reported in multiple instances within the document, and across different documents. This creates an opportunity for inconsistencies and a conflict if the model is created solely to serve generation of the document, i.e., when the document has been modelled rather than the system of interest. By reporting on a system model, which reflects the system of interest, the information is drawn from the same source in each instance it reported. An example in Figure 5 shows that components – conceptual in the case of the OCD - are described in section 5.1 of the document, and

the architectural hierarchy of those same components and depicted in section 5.2. A model-based, model-centric approach ensures that, if an element in the model is modified with updated project information, these changes will be reflected in all relevant sections (in this case OCD sections 5.1 & 5.2) when the document is generated. The document is not modelled *per se* – however the document template is used to establish the report mapping required to identify data elements, stored once in the date model, and consistently describe them multiple times throughout a document or document set.

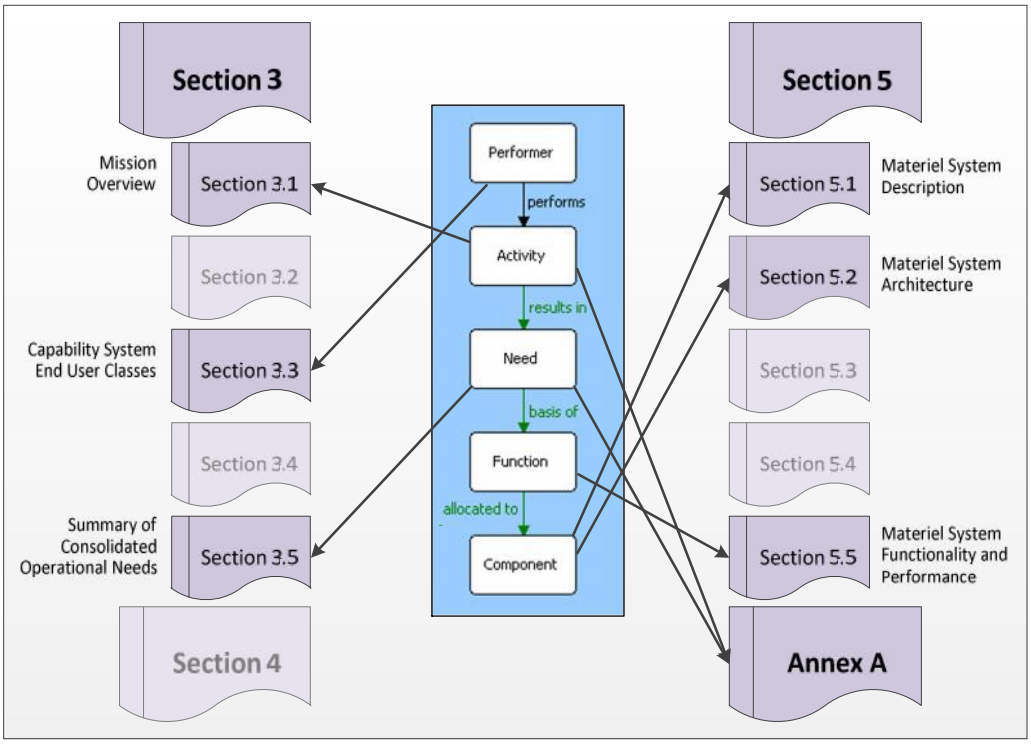

Figure 5. Data model to report document 'mapping'

Successful implementation of this example relies on a number of the benefits of the model-centric model-based approach derived from the focus on the information of and about the model, not on the required documents:

- **Traceability** – Links between pieces of information are established and maintained in the model-centric approach. These links allow the engineer to determine (to remember) what information is connect to other information, where an element came from. It also serves to facilitate impact analysis, by allowing the engineer to trace from an element under analysis to everything else it directly interacts with. The use of a document-centric approach only provides these advantages to the extent that the engineer fills in each text field, and then traces around through the document from section to section. Using this approach, the only way to maintain any semblance of link is manually.
- **Consistency** – Further to links allowing the source of other information and facilitating impact analysis, they also help to maintain consistency using the model-centric approach. If one element gets changed or removed, its impact upon other related elements is easily highlighted. The model-centric approach provides the further benefit that when the same piece of information is used in multiple places in the desired document output, it is drawn from the same source. When the document-centric approach is used, this changing information in one part of the document does not necessarily impact upon other related information. Also, when the same information appears multiple times in a document, it comes from independent sources, meaning that consistency needs to be manually maintained (by changing all sources at once if required).

- **Adaptability** – The use of the model-centric approach has great advantages in adaptability. As each document produced is merely a snapshot view of the information in the model at a moment in time, any number of these documents, or diagrams, can be produced 'fit for purpose' from this same information. The document-centric approach has the information ordered and structured as required for that particular document, or document set. From this individual view it can be difficult to create different related, 'fit for purpose' views for other interested stakeholders or project purposes during the system lifecycle.
- **Robustness and Information Sharing** – The use of a model-centric approach makes explicit the relationships between information to do with the system and its environment. This facilitates teamwork and continuity of projects, as these links no longer live solely in the individual systems engineer's head. Taking a document-centric approach, the relationships between information live within the systems engineers head and as such, true teamwork is difficult. This situation also makes projects vulnerable to schedule and technical difficulties occurring with changing project staff.

CONCLUSION

Systems engineering allows the design, implementation and support of systems that directly address user needs in a specified situation. Underpinning this approach is a holistic view of the system, interacting systems and environment. In the early days of systems engineering, systems engineers did their work with a system model in their head, with paper and pen for tool support. These supporting tools have developed over time, with the advent of word processors to replace the pen and paper and a further leap in supporting tools was the invention and development of software model-based systems engineering tools.

Model-Based Systems Engineering is the current growth area and way forward for future systems engineering. Modern MBSE tools allow the definition of complicated environments, systems of interest and interacting systems. The capability of these tools to define and track these interactions has gone beyond the ability of the human brain to consider. This advantage is particularly highlighted when systems engineering work is done in teams and a collective model of system is required. This paper has gone one step further in this move towards model-based systems engineering and examined the driving motivation behind the work – described as 'centricity'. This 'centricity' has been described as a spectrum, from entirely document-centric (focussed only on the products required for a project to continue through a committee process) to entirely model-centric (where all effort is put into defining a model of and about a system, then any required views or documents are produced as snapshot artefacts at any time). The centricity continuum applies both to the model-based approach as described and a purely document-based approach, with the focus on the document only to a focus on the model existing only in the systems engineer's brain. The emphasis of the systems engineer on the model of and about the system has advantages in traceability, consistency and facilitation of teamwork and continuity of information.

This discussion on a model-centric model-based approach in no way suggests that documents are a bad thing in the systems engineering process. Documents are the 'medium of the stakeholder' and allow information to be both gathered and disseminated in an effective manner. They are also a mainstay of many committee-based decision making processes. However documents are merely a chosen subset of not-necessarily-linked data associated with the system at any time. If you subvert the focus of the project, and merely model the required documents in a MBSE tool, the document is used to capture reality and the model is made of (and from) that document, thereby missing the dynamic cross linking and system focus of the model-centric approach. With appropriate tool support and planning, the model-centric approach can also maintain the communication benefits of the document by producing documents and other fit for purpose views directly from the system model as required.

REFERENCES

Crisp, H., *Systems Engineering Vision 2020*, Technical Operations INCOSE, Version 2.03, September 2007.

Department of Defence, "Data Item Description – Operational Concept Document – Version 1.4" (DID-ENG-OCD), Canberra, 2009.

Department of Defence, "Data Item Description – Operational Concept Document – Version 2.0" (DMSP(ENG)12-3-001), Canberra, 2011.

Estafan, J., "Survey of Model-Based Systems Engineering MBSE Methodologies" INCOSE MBSE Initiative, 2008, (http://www.omgsysml.org/MBSE_Methodology_Survey_RevB.pdf) Accessed 16 Aug 2011.

Haskins, C., *Systems Engineering Handbook: A Guide for Systems Cycle Processes and Activities*, SE Handbook Working Group INCOSE, Version 3.2.2, October 2011.

ISO, *Systems and software engineering – System life cycle processes*, ISO/IEC 15288:2008, March 2008.

ISO, *Systems and software engineering — Architecture description*, ISO/IEC/IEEE 42010:2011, November 2011.

Logan, P. & Harvey, D., "Architecting the Problem Space: an Architecture Framework-based Method for Definition of User Requirements", 4th Asia-Pacific Conference on Systems Engineering (APCOSE 2010), Keelung, Taiwan, October 2010.

Logan, P. & Harvey, D., "Documents as Information Artefacts in a Model Based Systems Engineering Methodology", 5th Asia-Pacific Conference on Systems Engineering (APCOSE 2011), Seoul, Korea, October 2011.

Metcalfe, R., "Packet Communication", Massachusetts Institute of Technology (MIT) Project MAC Technical Report MAC TR-114, December 1973.

Miller, G., "The Magical Number Seven, Plus or Minus Two Some Limits on Our Capacity for Processing Information", *Psychological Review* Vol. 101, No. 2, pp.343-352, May 1955.

Sparrow, B, Liu, J. & Wegner, D., "Google Effects on Memory: Cognitive Consequences of Having Information at Our Fingertips", *Science*: 333 (6043), pp.776-778, August 2011.

Waite, M. & Logan, P., "Model Based User Needs Analysis", Systems Engineering and Test and Evaluation Conference (SETE2011), Canberra, May 2011.

White, S., Alford, M., McCay, B., Oliver, D., Tully, C., Holtzmann, J., Kuehl, C., Owens, D. & Willey, A.,"Trends in Computer-Based Systems Engineering", IEEE International Conference on Computer Design on VLSI in Computers & Processors (ICCD '92), Washington, DC, USA, 1991.

BIOGRAPHY

David Harvey - Dr David Harvey is a systems engineer with a particular interest in Model Based Systems Engineering (MBSE). He holds a bachelor degree and a doctorate both in the field of mechatronics. He currently leads the MBSE program at Aerospace Concepts Pty Ltd. This team is developing an MBSE approach and tailored tool to assist in complex system definition in conjunction with Australian Defence partners. As well as this development, he is also involved in applying the tool and approach to capability definition in major Australian Defence projects.

Michael Waite - Michael has been working as a professional engineer for over nine years since completing his Bachelor of Engineering (Mechatronics) degree in 2001. His career has seen him working for several multi-national automotive companies in Australia, Asia and Europe, including Mitsubishi Motors, Ford and Caterpillar. He currently works for Aerospace Concepts, a systems engineering consulting company, specialising in the development of complex-system capabilities.

Paul Logan - Following a 23-year military career, Paul Logan has acquired more than 20 years of

experience working in the Australian defence and commercial communications sectors, focusing on requirements engineering, system and enterprise architecting, and Model Based Systems Engineering (MBSE). Paul has gained Master degrees in Information Science and Professional Accounting, bachelor degree in Communications Engineering, and postgraduate qualifications in Management, Finance and Investment. Paul works as an independent consultant, and also as an instructor of MBSE methods and tools. He continues to pursue his interest in systems engineering theory and practice applied to capability and product definition.

Tommie Liddy - Tommie Liddy is a mechatronic engineer completing his Ph.D. in Robotics at the University of Adelaide while working as part of the Model Based Systems Engineering (MBSE) team at Aerospace Concepts. His academic study has focused on navigation control for Ackermann vehicles and uses vector fields as control schemes. Development of this work was achieved through simulation of vital concepts then a physical implementation of the final system. As part of the MBSE team at Aerospace Concepts Tommie is developing MBSE tools for operational analysis and capability definition.