Model Based User Needs Analysis

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

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

Abstract. Most descriptions of the systems engineering process commence with words or a pictograph indicating "receipt of customer requirements". The process descriptions then go on to describe activities for system requirements analysis and derivation as the baseline for system development. This view of systems engineering is a system-centric rather than user-centric view and is perhaps a correct implementation of system engineering from the developer's perspective. But from where do the customer requirements come and how are they developed? While this is perhaps the realm of the business analyst rather than the systems engineer, increasingly systems engineers are challenged with development of the overall user capability rather than just the technology implementation. Systems engineers are now required to apply their proven system requirements analysis techniques to the generation of user needs. This paper describes a model-based systems engineering methodology for the analysis and generation of user needs (and constraints) and provides an illustrative example of its application.

INTRODUCTION

Numerous definitions of systems engineering exist and a greater number of process descriptions can be found (INCOSE 2010, Ryan and Faulconbridge 2003, Martin 1997). The International Council on Systems Engineering provides a definition that states that "systems engineering is an interdisciplinary approach and means to enable the realization of successful systems" (INCOSE 2010). The definition continues to describe systems engineering as "defining customer needs and required functionality early in the development cycle … and then proceeding with design synthesis and system validation … while considering [all aspects of the system life cycle]." The description ends with the statement that systems engineering "considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the users needs".

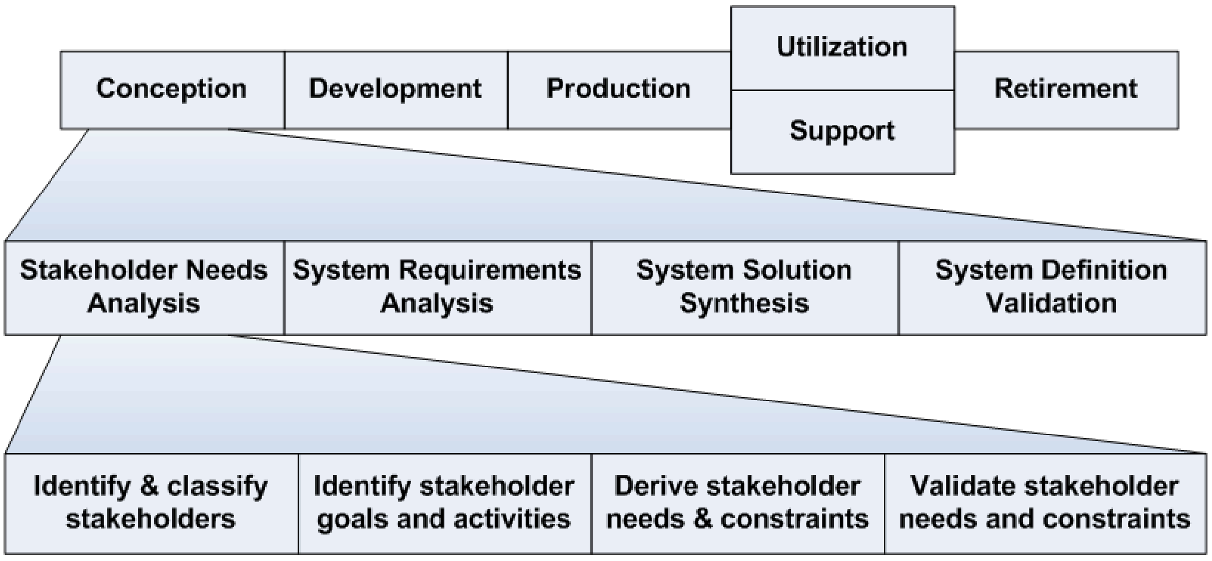
Conception
Development
Production
Utilization
Support
Retirement
Stakeholder Needs Analysis
System Requirements Analysis
System Solution Synthesis
System Definition Validation
Identify & classify stakeholders
Identify stakeholder goals and activities
Derive stakeholder needs & constraints
Validate stakeholder needs and constraints

Figure 1 System lifecycle stages and activities

While "life cycles vary according to the nature, purpose, use and prevailing circumstances of the system" (ISO/IEC 2008), a basic depiction of the system life cycle consistent with the above definition is shown in Figure 1. The more generic term *stakeholder* is preferred to *customer* and requirements analysis identified as the means to describe (among other things) system functionality.

The initial stage of the system life cycle is identified here as *Conception.* Developmental activities performed in this foundational stage first identify and refine the user and other stakeholders' needs that a valid system solution must address. Then activities follow that analyse the identified needs and derive system requirements which any system solution must verifiably meet. The specified requirements are then addressed by the optimal selection of a particular system from among a number of feasible alternatives.

The very foundation of system development and the basis for success of the system throughout its life is the identification and refinement of user and other stakeholder needs. The product of this essential and critical system development activity, commonly referred to as a *User Requirement Document*, clearly must be complete, consistent, coherent, and precise. Yet it is recognised that this is perhaps the system development activity most fraught with risk and most often poorly done, if done at all (INCOSE 2010). The difficulty exists due to the inherent human nature of the task – most stakeholders are people with all the human frailties involved in succinctly forming and committing to one or more statements of need.

Model-based methods have evolved in the software development domain to deal with the difficulty involved in cogent definition of software requirements and software design (OMG 2011). The definition and development paradigm is now being extended into the system domain as model-based systems engineering (MBSE) (INCOSE 2007). Model-based systems engineering is the formalised application of modelling 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 2010).

While the INCOSE description of MBSE states that MBSE commences in the conceptual design phase and identifies support for system requirements activities as one of its applications, most process and methodology descriptions are focused on functional analysis of user needs to derive and develop the systems requirements; leaving generation of user needs to the vagaries of *soft systems science* and non-model-based methods. While much of user needs elicitation does involve soft systems techniques, the proven systems-thinking and modelling methods and techniques that underpin MBSE elsewhere in the system development life cycle can be applied to the identification and definition of user needs.

BUSINESS ANALYSIS AND SYSTEMS ENGINEERING

The International Institute of Business Analysis (IIBA) describes the role of *business analyst* as: "the set of tasks and techniques used to work as a liaison among stakeholders in order to understand the structure, policies, and operations of an organization, and to recommend solutions that enable the organization to achieve its goals" (IIBA 2011). This is a role that most systems engineers, or at least the requirements engineers, working in needs analysis and user needs definition would ascribe to themselves. If "solutions that enable the organization to achieve its goals" is interpreted as identifying the capabilities that the organisation (i.e. the stakeholders) need to achieve their collective objectives, then the system engineer is doing *business analysis*, the product of which is a consolidated statement of user requirements.

This seemingly confused situation simply highlights that the identification and definition of user and other stakeholder needs addresses a broad spectrum of domains and disciplines. It also reflects the diversity of systems engineering roles and tasks throughout the life cycle either during development or throughout the operational life of the system (Emes *et al* 2005). The system engineer/ business analyst referred to here is, or should be, a small multi-disciplinary team (INCOSE 2010). What the system engineer, skilled in model-based techniques, specifically brings to business analysis and stakeholder needs definition is system analysis techniques long familiar in definition of the solution applied to definition of the problem. Modelling the problem space and bringing a rigorous framework to identification and definition of user and other stakeholder needs can be considered to be "architecting the problem space" (Logan and Harvey APCOSE 2010).

The problem space is the context of the system of interest. The problem that stakeholders have a ‘need’ to address exists in that conceptual and physical space into which the system – the solution system – will be deployed. The problem space is the domain of the stakeholders (i.e. the ‘business’ or ‘enterprise’), a domain that must be fully understood by the system engineer/business analyst if the problem is to be adequately described, such description enabling the requirements for the solution system to be effectively defined (Martin 2004). Recognising that the users’ domain/problem space is and can be modelled as a system allows all the model-based systems engineering methods, techniques and tools to be used in what has been previously considered to be an inevitability poorly structured process.

A question arises: "Why are the systems engineers developing the customer requirements?” Description of the stakeholder environment, identification of their business goals and objectives, and statement of their needs and desires is the realm of the business analyst. Yet systems engineers are increasingly doing, involved or laying claim to this life cycle stage (INCOSE 2010).

The model is a system in and of itself. The system definition and design skills together with systems engineering management procedures such as requirements, data and configuration control are required to ensure the model itself is managed and controlled in a manner that establishes and preserves model integrity while achieving the aim of the modelling activity – a classic system engineering challenge. There exists a need to ‘do systems engineering to practice systems engineering’. The system engineer becomes the business analyst of the model; to then avoid the model becoming the product rather than a representation of the system, the (real) business analyst becomes a stakeholder in the model – the bridge between the modelling systems engineer and the real stakeholders; the end-users of the model and the system that it represents. Some individuals with requisite breadth and depth of knowledge and experience in both systems engineer and the relevant business domain can act in both roles.

“Essentially, all models are wrong, but some are useful” (Box and Draper 1987). When the model is treated as a system and developed using the same procedures and methods that the systems engineer applies to definition and development of the system of interest, then the model will be much less wrong and much more useful.

STAKEHOLDER NEEDS AND SYSTEM REQUIREMENTS

Stakeholder needs occur when there is a problem in their environment that one or more of the influential stakeholders decide to address. The problem space is the aggregate of various stakeholder environments as viewed by that stakeholder, or more specifically, class of stakeholder. The problem space is thus a multi-element entity consisting of a range of stakeholder elements that interact in some manner to achieve both individual and collective outcomes. The problem space, the context in which the system of inertest must function, therefore comprises a system as generally defined and described and this can be modelled in the same manner as the solution system. Each stakeholder then has needs or constraints on the systems of inertest and the system then must function and address the constraints in a specific manner to satisfy concurrently the stakeholder needs. Stakeholder needs are therefore identified and described per stakeholder while system requirements are identified and described by system function or constraint. It is this difference in points of reference that is the fundamental distinction between user needs and systems requirements.

The purpose of the problem space model is to facilitate elicitation of stakeholder needs by identifying what outcomes a stakeholder seeks and how the stakeholder intends to achieve that outcome. The actions that each end-user performs in pursuit of one or more goals are captured. Also captured are non-user stakeholder business rules imposed on users and also the limitations those stakeholders place upon any solution. This focus, at the outset of stakeholder needs definition and before derivation of system requirements, facilitates capture and definition of a coherent and consistent set of both the stakeholder needs and subsequent system requirements.

In Figure 1, significant and deliberate distinction is made between stakeholder needs analysis and system requirements analysis - distinct stages of the system lifecycle are identified. In practice the stages are more concurrent and interrelated, but experience reveals user and other stakeholder needs must be robustly addressed before significant work on derivation of system requirements occurs if those requirements are to be a sound platform for subsequent system definition and development.

User needs and other stakeholder requirements are identified and described from the perspective of a particular class of stakeholder. Each of the stakeholder-class perspectives is addressed by modelling each stakeholder class and their environment with emphasis on identifying what are their concerns with the system of interest - i.e. what they need to achieve (goals and objectives), or to what they need the system to be or not to be when developed to conform (limitations and constraints). The aggregate model of all stakeholders is thus an integrated architecture description of the problem space (ISO42010 2008). That architecture description identifies the entire set of stakeholder classes and their respective concerns - i.e. their needs and constraints on the system of interest grouped by shared concerns.

Once stakeholder needs are captured and consolidated by end-user and other stakeholder class, they can be functionally analysed to determine what the system must do to address the needs. This gives rise to a simple algorithm: *if the **user/stakeholder** needs to do that, then **the system** shall do this* (i.e. exhibit a function) *or be this* (conform to a constraint).

ELICITING AND ANALYSING NEEDS

In order to elicit the user needs the operational context must first be established, the first step of which is to identify the users. (ISO15288 2008) describes a user as an “individual or group that benefits from a system during its utilization”. Once the users have been identified (“who” is involved), the scenario in which they operate is defined (“where” and “when”). The systems engineer, in collaboration with the stakeholders, develops representative scenario(s) which describes a typical use “day-in-the-life” of the system of interest. The Australian Defence Capability Definition Documents guide, section 5.2.2.1 states the set of scenarios “…should be as small as possible but needs to span the full functionality and maximum performance requirements of the capability system from the [end-user] perspective”(ADO 2009). The scenarios need not cover every activity and task that the system of interest is to be employed, more so, the scenarios should cover *enough* of the problem space such that the relevant needs of the system can be elicited.

From the situation identified in each scenario, the systems engineer can model the operational activities which represent the actors in the scenario undertaking activities according to their role. Functional Flow-Block Diagrams (FFBDs) are commonly used for activity modelling as the FFBDs are quickly generated (and modified) and easy to read, particularly by non-technical end-users and business-orientated stakeholders. By utilising a model-based approach, the analyst can ‘walk’ through the FFBDs in the model and allow the subject-matter experts to focus their attention on each operational activity (the discrete elements of the business process). As each activity is individually considered, the needs are identified and are recorded directly into the model.

Complex systems have complex behaviours and as such require complex behaviour diagrams. To simplify the diagram, and increase readability, activity decomposition is employed. This results in some activities existing as an aggregation of several low-level activities. When walking through the model, the needs elicitation process is done at the level of decomposition at which decomposing further would offer no further needs. This is often at the leaf level (i.e. lowest level of decomposition) of the model. If, however, every activity within an activity decomposition shares a need, this need will be associated with the higher-level activity which has this common need.

The needs elicitation process involves identifying and recording the needs of the users of a system. Ideally, this should be a complete, consistent and relevant set of needs which fully express the goals and objectives of the users of the system. However, in practice we find that users identify ‘wants’ and solution designs termed as needs as well as genuine needs. This is to be expected and does not raise a substantial issue as analysis will filter out such extraneous needs.

The method for clarifying solution-independent needs is to apply a similar process to the ‘Five Whys’ technique of problem solving, developed by Sakichi Toyoda, founder of Toyota Industries Co. Ltd. By asking “Why?” repeatedly (not necessarily five times) it is often possible to drill down to the underlying need behind what the user has expressed, as shown in the example in Figure 2.

Salesman: “I need a notebook computer!”

Technical Manager: “I am sure you do, but why (do you need a notebook computer)?”

Salesman: “Because I need to use a computer when I am with a prospective customer.”

Technical Manager: “Why (do you need to use a computer when you are with a prospective customer)?”

Salesman: “Because I need to prepare a sales proposal when I am with the customer.”

Technical Manager: “Why (do you need to prepare the proposal when you are with the customer)?”

Salesman: “Because when I return to the office to produce the proposal [as I do now], I often lose the sale … because the customer goes cold or my competitors get in there.”

Technical Manager: “Ah … you're learning but why do you prepare the proposal in the office?”

Salesman: “I need up to date prices and product details and the sales system verifies the proposal”.

Technical Manager: “So, you need the means to access update data and produce a verified sales proposal when with a prospective customer?”

Salesman: “Yes! That's what I said – I need a notebook computer! And, oh – I need it to have mobile broadband! And, I need a portable printer to print the proposal!”

Technical Manager: “Mm – perhaps you need to have continuous access to current sales and product data to prepare and deliver a verified sales proposal when with a customer (to avoid the customer going cold and preventing competitors from stealing the sale.)”

Salesman: “Ah – more or less …”

Technical Manager: “So, the solution maybe an application on your 4G mobile phone (which I bought for you last month) which connects to the sales computer in the office and you can email a pdf version of the proposal to the customer and they can print it in their office”.

Salesman: “Ah … maybe - but what I want is a notebook computer!”

Figure 2 – Applying the ‘Five Whys’ technique to elicit user needs

The user (the Salesman) instead of needing “a notebook computer” (a solution), actually needs “to have continuous access to current sales and product data to prepare and deliver a verified sales proposal when with a customer”. The level of this user need would be detailed in an associated measure of effectiveness (MOE) and would be dependent upon the scenario that is being analysed to aid in elicitation of the needs.

Three types of needs are likely to be encountered during the needs gathering phase: conscious user needs, unconscious user needs and undreamed-of user needs (Robertson and Robertson, 1999). The conscious user needs are relatively easy to capture, as they are at the forefront of the user’s mind and / or it is immediately obvious that a capability gap exists. The unconscious needs however are more difficult to elicit as the expert understands these needs so well that it seems too obvious to them to identify. These needs are often overlooked when the SMEs know a lot about their area of expertise and assume everyone else has the same knowledge. Undreamed-of needs however are needs that the user is unaware of the possibility of existing; this may be due to lack of technological expertise or perhaps not having considered using the functionality of the system other than the prescribed manner.

The two most important aspects of user need elicitation are effective communication with the stakeholders to capture needs; and traceable and verifiable management of those needs once captured.

MODEL-BASED METHODS

‘Models’ *per se* are not new, whereas making the development ‘model-based’ is relatively new. Models as representations of an entity yet to be realised have long been used to facilitate communication: civil architects use scale models, as do naval architects and vehicle designers, and ‘mud maps’ have been used throughout the ages by tactical field commanders to ensure understanding of their intent. Nonetheless, formal agreement and specification in Systems Engineering continues to be represented in written documents containing mainly text and two dimensional drawings.

‘Diagramming’, or ‘drawing’, is different to ‘modelling’. Drawing is a two dimensional paradigm (notwithstanding the use of isometric techniques to represent solid objects). It seeks to communicate and achieve understanding by representing details of things in a diagrammatic or pictographic fashion. Diagrams serve two basic purposes:

- facilitate ease of understanding - visual representations are often more readily understood by stakeholders than descriptions in text; and
- facilitate communication of complex information – “a picture is worth a thousand words”.

Modelling differs from diagramming, in that it is multi-dimensional in nature. Whereas diagrammatic representations generally display information in two dimensions, modelling allows basic information about an entity or interest to be enriched with relevant data that is gathered or developed as part of the modelling process. The entity-relationship nature of modelling means that information about the system can be interlinked forming a multi-dimensional model - diagrams are a slice through the model. Improved technology and tools have made the collection and processing of an information rich model far more achievable. The tools also enable multiple dimensions/information sets to be collected and stored as well as allowing the production of two dimensional maps – the diagrams which depict the entity of interest. But now the diagrams (and other useful descriptions) are views on the model, rather than being isolated representations of aspects of the system.

Due to the inter-related nature of a model-based approach, experts in specific areas can concentrate their expertise in relevant parts of the model, and when this information is linked correctly it influences other related areas within the model. The output from the scenario-based needs elicitation process, the operational needs, can be recorded directly into the model during the elicitation process. This model can then be further developed in subsequent phases of the systems engineering process.

AN EXAMPLE – A FANCY RESTAURANT

To illustrate these ideas, we have considered a model-based requirements analysis for a restaurant. In this case we have a fine-dining capability where the restaurant staff use the restaurant system as shown in Figure 3 in order to satisfy the needs of the customers.

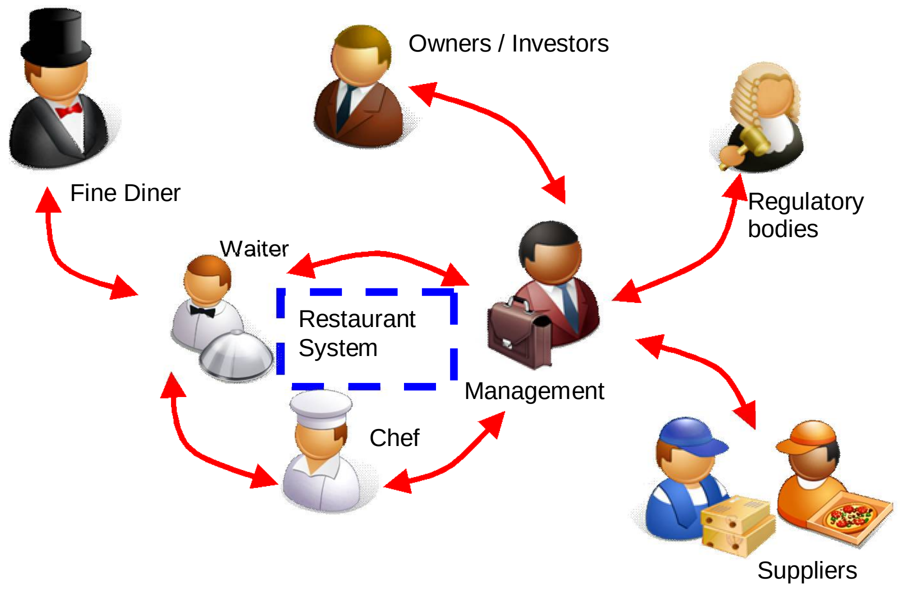

Figure 3 – High-level operational context diagram for the fine dining restaurant

In addition to the customers, the Fine-dining Capability’s users also include the owners/investors (whose need, no doubt, is to make a return on their investment, with an associated financial MOE), the restaurant suppliers (who needs include order-tracking, invoicing, etc) and regulatory bodies who add constraints to the system (zoning, health standards, taxes, etc). For our example we will consider only the customer: the fine-diner.

Applying the model-based approach to the needs analysis starts with creating a scenario-based model of the activities of the users; in our example a customer visits our restaurant and orders a meal and, once finished, pays the bill (see Figure 4).

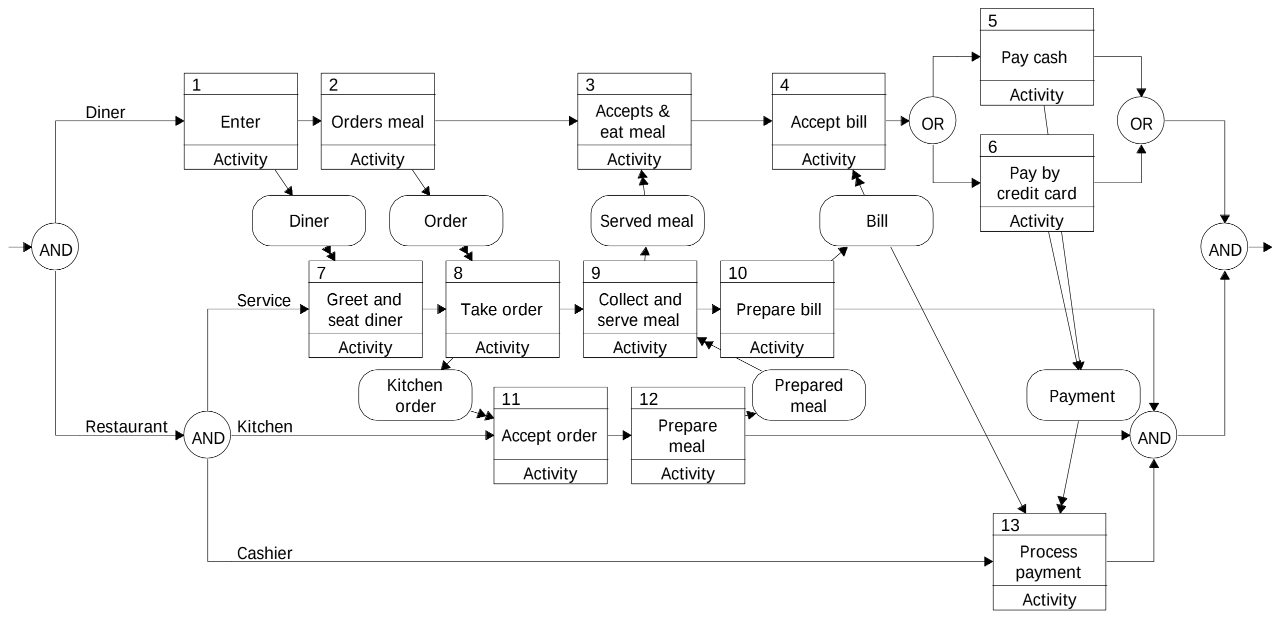

Figure 4 – Operational activity model for user needs elicitation

By consulting with a typical user, someone with fine-dining experience perhaps, the activity model for this scenario can be generated. The advantage of using a model-based analysis become apparent when, due to its plastic nature, the model is constantly moulded and shaped into a representative view of the activities involved in the scenario, including information and resource transfer, sequencing and concurrency of actions and so forth. This process may be repeated several times for each of the different user classes to ensure that the activities are suitably modelled.

Once the activity model is accepted, the systems engineer can 'walk through' the model and at each activity ask the users / stakeholders / domain experts "what needs does this user, acting in this role, have, performing this activity, in this scenario?" In the figure above, looking at activity 2 "Orders meal" the question could be asked "What need does the customer (user) have when acting as a fine-diner (role), whilst ordering a meal (activity) during a visit to a fancy restaurant (scenario).

CONCLUSION

Clear and complete definition of system requirements is necessary to ensure that the developed system solution can be tested and verified as meeting the stated requirements that is the system has been built *right*. This however does not necessarily ensure that the *right* system has been developed, that is, the system when used will satisfy the needs of not only end users but also the other stakeholders with an interest in the outcome. Clear, complete, and consolidated definition of the needs of all stakeholders is the essential foundation for effective system development and ultimate success.

The key to successful stakeholder needs identification and system requirements definition is recognition of the relationship between the problem and solution space. The problem and hence the stakeholder needs lie in the context space of the solution. Stakeholder needs can be derived from analysis of the system context – before any decisions are made as to the definition of the system and its requirements.

Model-based needs analysis involves development of a model of the stakeholders' collective environment. The stakeholders are identified, their relationships captured together with their goals, their activities in the pursuit of those goals are mapped and then analysed to derive their needs and constraints on the system of interest. The resultant stakeholder model constitutes an architecture description of the stakeholder enterprise. The proven methods, techniques and tools of enterprise architecting and system modelling can be brought to bear on what is often regarded as a poorly defined and ambiguous activity, the elicitation of stakeholder needs. Model-based systems engineering methods bring structure and rigour to the socio-analytic skills of the business analyst.

REFERENCES

Australian Defence Organisation (ADO), *The Australian Defence Capability Definition Documents Guide v1.4*, Canberra, 2009

Box, G. and Draper, N, *Empirical Model-Building and Response Surfaces*. Wiley, 1987

Emes, M., Smith, A. and Cowper, D., *Confronting an identity crisis—How to "brand" systems engineering. Systems Engineering*, 8: 164–186. doi: 10.1002/sys.20028, John Wiley and Sons Ltd. Chichester, UK, 2005

Faulconbridge, R and Ryan, M., *Engineering A System: Managing Complex Technical Projects*, Artech House, Norwood 2003

International Institute of Business Analysis (IIBA), *What is a business analyst?*, viewed 31 January 2011 <http://www.theiiba.org/AM/Template.cfm?Section=What's a BA>

International Council on Systems Engineering (INCOSE) SE Handbook Working Group, *Systems Engineering Handbook* v3.2, INCOSE, San Diego, 2010

International Council on Systems Engineering (INCOSE) *Survey of Model-Based Systems Engineering (MBSE) Methodologies*, INCOSE-TP-2004-004-02, Version 2.03, INCOSE, September 2007

ISO/IEC 42010:2007 *Systems and software engineering -- Recommended practice for architectural description of software-intensive system*, International Standards Organisation, 2007

ISO/IEC 15288:2008 *Systems and software engineering -- System life cycle processes International Standards Organisation*, 2008

Logan, P. and Harvey, D, Architecting *the Problem Space*, Proceedings of APCOSE10, Taipei, 2010.

Martin, J, *Systems Engineering Guidebook*, CRC Press, Boca Raton, 1997

Martin, J, *The Seven Samurai of Systems Engineering: Dealing with the Complexity of 7 Interrelated Systems*, Proceedings of the INCOSE Symposium, 2004.

The Object Management Group (OMG), *OMG Model Driven Architecture*, viewed 25 June 2010, <http://www.omg.org/mda>, 2010.

Robertson, S. and Robertson, J., *Mastering the Requirements Process*, Addison-Wesley, Boston, 1999

BIOGRAPHY

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.

Following a twenty-three career in the Australian Army, Paul Logan acquired twenty years experience with model-based systems engineering methods, techniques and tools. He introduced MBSE into the Jindalee Operational Radar Network project in 1991 and has since applied model-based analysis and design in commercial and military projects. From 2002 Paul has been involved in Capability Definition Document (CDD) development for the Defence Department. Paul is a certified instructor of Vitech Corporation's introductory and advanced courses on Model Based Systems Engineering using CORE®. Paul holds Bachelor of Engineering (Communications) and Master of .Information Science degrees. He is a member of INCOSE, IEEE and SESA, of which he is the current President.