Model-Based Conceptual Design through to system implementation – Lessons from a structured yet agile approach

Matthew Wylie
Shoal Engineering Pty Ltd
matthew.wylie@shoalgroup.com

Dr David Harvey
Shoal Engineering Pty Ltd
david.harvey@shoalgroup.com

Tommie Liddy
Shoal Engineering Pty Ltd
tommie.liddy@shoalgroup.com

ABSTRACT

This paper discusses the successful application of model-based conceptual design, followed by Agile Scrum software development. Model-based conceptual design links information about the understanding of a problem to possible solutions. Although this technique can provide a richness of information and explanation that is difficult to capture through "flat" views on the information (such as specification documents), it is often difficult to provide and utilise this richness beyond the conceptual phase into the design and production stages. In this project, a model-based approach was used to support initial concept exploration through problem definition, decision to build a new system and production of a solution class specification. System design was then conducted according to this specification and supporting information in the model – bringing with it some of the benefits of this model-based approach. An Agile Scrum approach to software development was used in the design and implementation of this technical solution. This combined model-based conceptual design and Agile Scrum development approach uncovered some challenges and many benefits. The project retrospective analysis also revealed some potential process enhancements for consideration in future projects. These potential enhancements include increased software development team involvement in the conceptual design phase, further application of the Scrum framework to model-based conceptual design, and employing a fully integrated conceptual design and software development agile development approach.

INTRODUCTION

This paper uses the case study of the development of a weapons range safety planning tool to examine how a system can be defined using model-based conceptual design, and then realised through Agile Scrum development. The paper will use the case study to discuss how the richness and completeness of the system model was utilised to enable agile development in the delivery of an assured safety-critical system, as well as the lessons learnt and potential process enhancements for employment on future projects.

CASE STUDY

Weapon range safety planning is conducted prior to live firing exercises to determine safety exclusion zones and ensure the safety of people and facilities. Figure 1 provides an example of a weapon range safety template. In many cases range safety planning is still based on manual methods and is achieved using pencil, paper, look-up tables, and tried and tested processes. With the increased use of automated support in Defence, a project was started to develop a software tool to aid range safety planning.

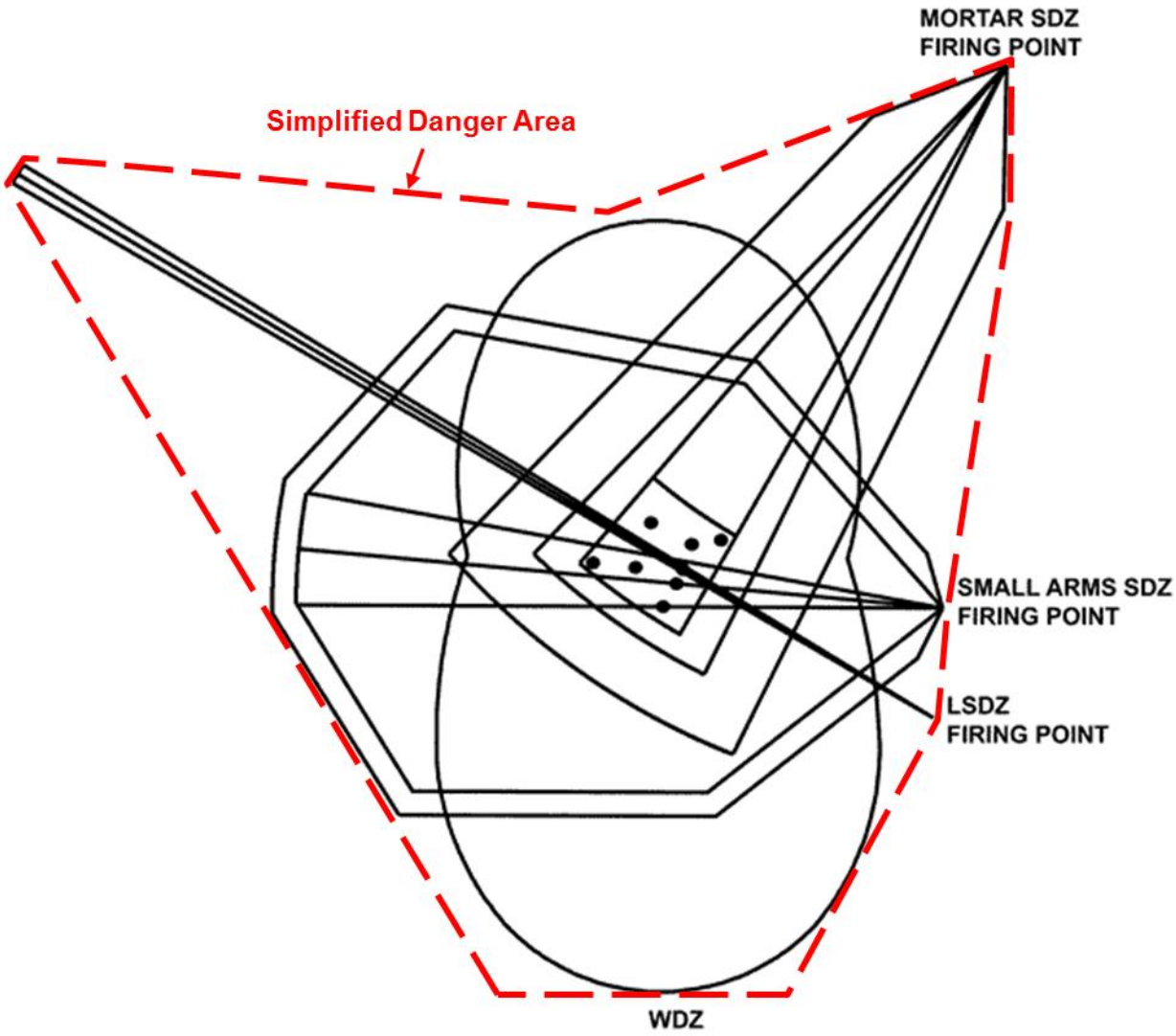

Figure 1. Example US military range safety trace. An overlay has been added to show how a simplified Danger Area geometry (broken lines) can be extracted from the complex intersection of plotted danger areas (solid lines).

CONCEPTUAL DESIGN UTILISING MODEL-BASED SYSTEMS ENGINEERING

Model Based Systems Engineering (MBSE) tools and methods, in the form of the Whole of System Analytical Framework (WSAF), were used in the capability definition for this system. The WSAF was developed by the Defence Science and Technology Group to aid modelling, simulation and analysis of complex capabilities, and was then further extended to support capability development (Robinson and Graham 2010, and Robinson et al. 2010). It enables an organised, traceable architectural model of a complex capability to be created and utilised for knowledge capture and definition of that capability. It focuses on using a structured approach to explore and define the problem space to ensure that a capability will address the right problem (Logan and Harvey 2010). The WSAF also provides processes and tools for the generation of system views and documentation for use in down-stream development activities. Example views are shown in Figure 2 and Figure 3. These tools and methods aided in the development of a complete and comprehensive conceptual design. This approach also presented an opportunity to exploit the similarities between MBSE methods and software development methods, particularly around the use of functional models and decompositions as a source of

requirements.

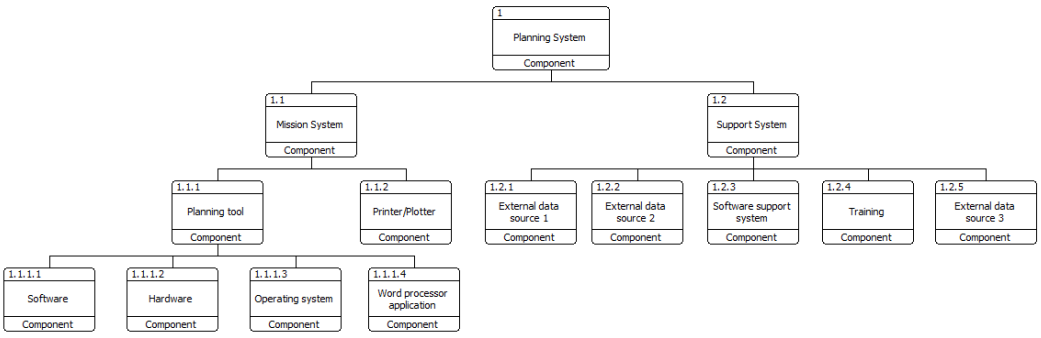

Figure 2. Example model view - System Hierarchy

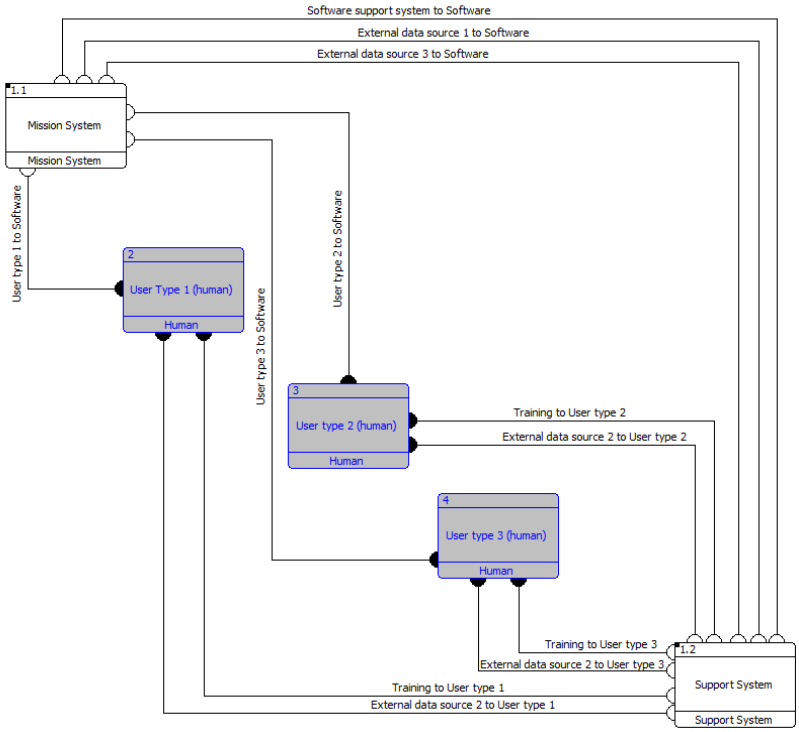

Figure 3. Example model view - System Connectivity

The capability definition for the system was accomplished using a spiral development approach starting with an examination of guidance policy and doctrine, undertaking a scenario based needs analysis, modelling existing systems and determining solution independent constraints. This was followed by the definition and specification of the solution system over several iterative concept development cycles with knowledge captured within a CORE model. Key elements of the model included, but were not limited to: system architecture and components, functional models and corresponding functional requirements, interfaces and links, and system constraints.

To meet the assurance requirements for this safety-critical system, a hazard analysis was also undertaken as part of the conceptual design process. Hazard mitigations were built into the solution system design and affected model elements identified.

SOFTWARE DEVELOPMENT UTILISING AGILE SCRUM

Following the MBSE conceptual design activity, an agile scrum software development process was used to develop the software products. This approach uses iterative software development, using a series of development 'sprints' for the incremental delivery of functioning products. It allows for adaption to change during the development process, and ensures a continued emphasis on collaboration with the stakeholder team towards an appropriate system solution (Schwaber and Sutherland 2013).

As described in the Agile Manifesto (Agile Alliance 2001), agile places more value on individuals and interactions, customer collaboration, working software and responding to change when compared to traditional development methodologies. This creates an adaptive rather than predictive development environment (Boehm and Turner 2003).

While agile development methods were first seen as incompatible with traditional plan-driven, process-based methodologies, a balance of agility and discipline can be employed to leverage the benefits of agile whilst managing risk (Boehm and Turner 2003).

MODEL TRACEABILTY

The MBSE approach to capability design provides a highly structured and traceable system design where system requirements can be linked back to functionality, user needs, use cases, and high-level guidance. An example of such traceability is shown in Figure 4. A model also provided a mechanism for capturing and understanding the more complex connectivity between elements of the capability.

Figure 5 shows a subset of the WSAF model element classes and relationships utilised in the conceptual design activity for the range safety software tool. The richness of information in the model was utilised to add context and detail to the requirements set and system specification. Specific views and diagrams were able to be generated from the WSAF model to assist the customer's understanding of the system design. By utilising this model-based approach to capability design, the solution system functions were already defined and modelled prior to the software development phase. Additionally, the system requirements were already organised by function and could be traced back through the problem space analysis to the original driving guidance. This reduced the software development effort normally required to translate system requirements into functional software design specifications, and reduced the potential for introducing errors through the misinterpretation of system requirements by the software development team.

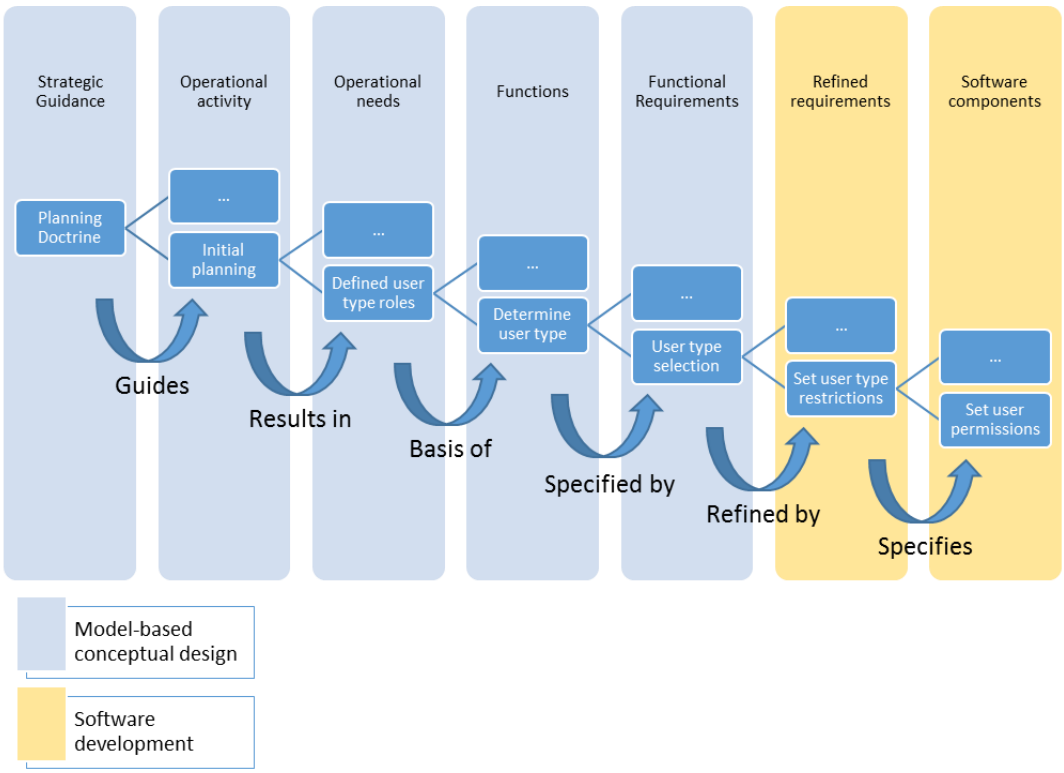

Figure 4. Traceability example

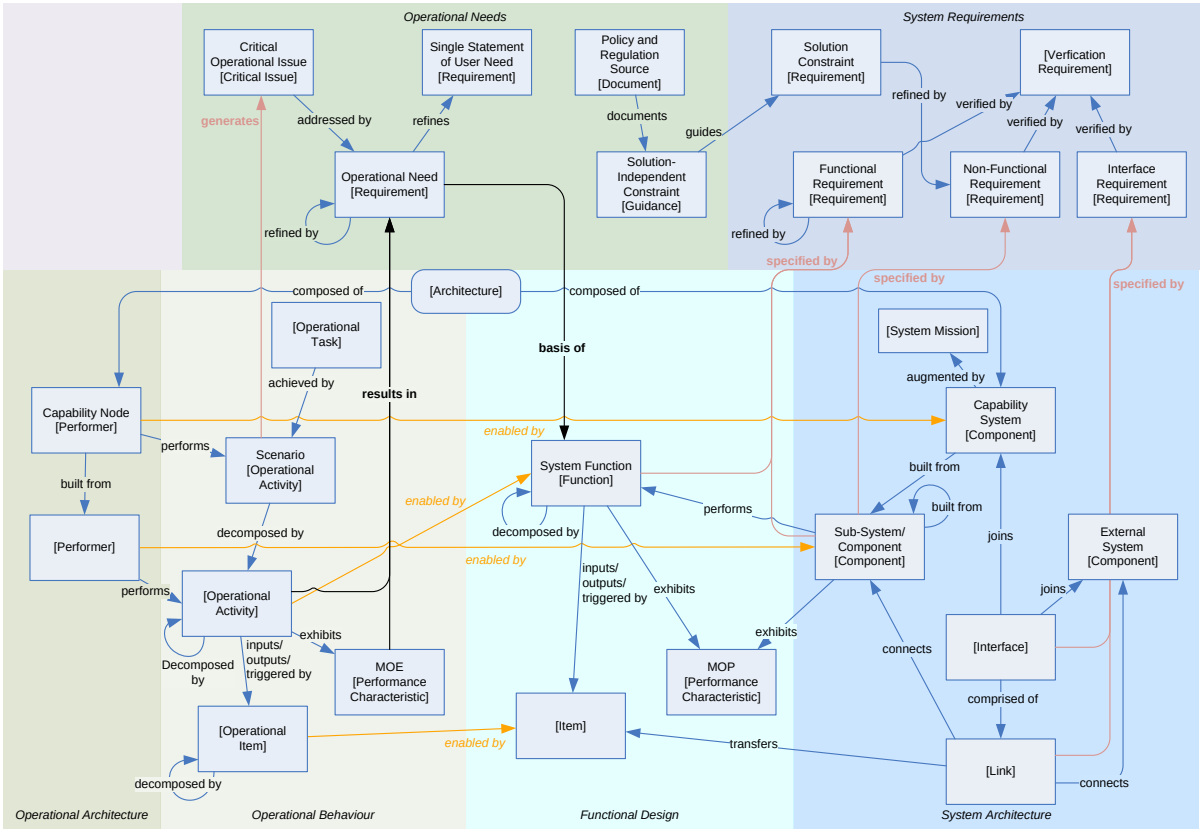

Figure 5. A subset of the WSAF model classes and relationships utilised in the conceptual design

COMBINING MODEL-BASED CONCEPTUAL DESIGN AND AGILE DEVELOPMENT

Utilising Agile software development methods for the development of complex, software-reliant systems provides significant advantages in providing a flexible development environment that embraces change and allows for evolving stakeholder needs to be addressed. However, where systems are required to meet defined levels of assurance and performance, a strong underlying architecture is also required. In this case, an agile development approach must be supported by an architecture that provides flexibility, while maintaining the required integrity for an assured system (Boehm 2010).

Employing an agile scrum software development method allows for design flexibility during development and prioritisation of the features of highest importance to the customer. The regular delivery of functioning software products at the conclusion of each sprint, and enhanced collaboration between the stakeholder and development teams, enables early customer evaluation and feedback.

Where design changes are required they can be represented in the model to understand their impact on the overall system. Figure 6 demonstrates how the model-based traceability can be linked into the software development domain to provide ongoing system design guidance during product development.

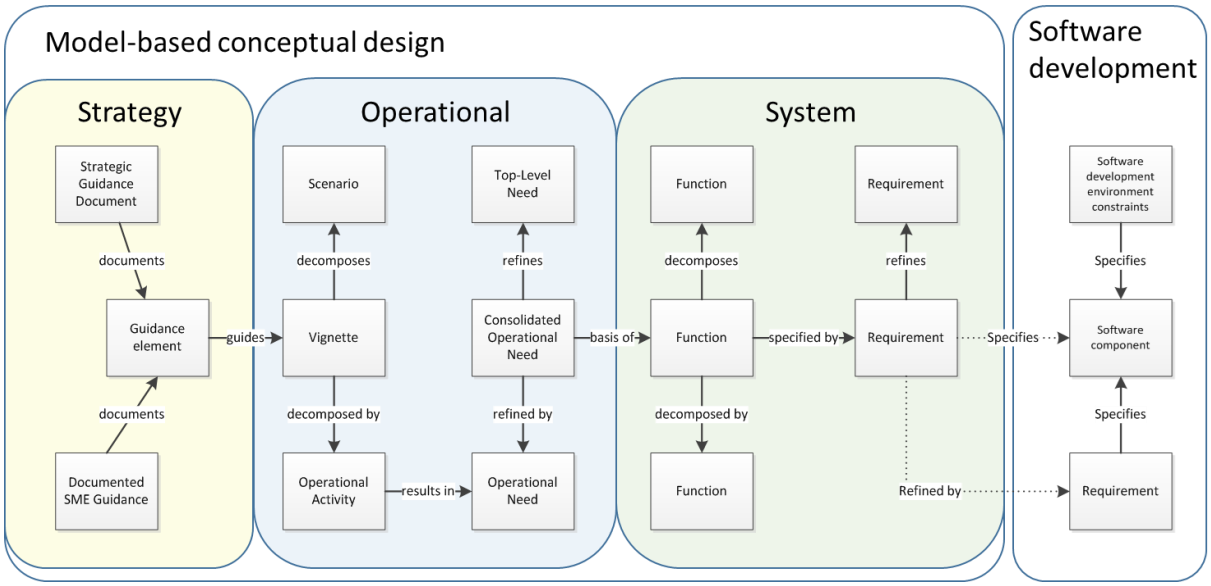

Figure 6. Traceability from conceptual design into software development

Utilising an MBCD approach for system definition provides an underlying system architecture that can efficiently support changes resulting from use of agile software development methods. The model can be readily utilised to identify the wider system elements affected by a change through examination of the directly affected elements and their relationships to other model elements. For example, a proposed change to system functionality, the model can first be used to identify the specific functional elements of the system that are changing. This set of functional elements will be linked in the model to other element types such as components, operational needs, functional requirements, resources, triggers, input artefacts and output artefacts. The linkages will identify those elements which may be affected by the change allowing the user to systematically consider the impact to those linked elements. Any impact or required change to a linked element can then be analysed in the same way (e.g. an interface of a component), and this analysis continued to the depth required for the effects of the change to have been adequately considered.

The flexibility of the agile scrum approach and the understanding of the impact on the system gained through the WSAF model allowed design changes to be implemented earlier in the development process than would normally be achieved through sequential development methods.

The combination of these methods ensured that the project had a focus on understanding the client need and solving the right problem throughout problem definition, solution specification and solution system implementation. This should increase the likelihood that not only is the solution verified and meets requirements, but that it is a valid solution and addresses the client problem.

LESSONS LEARNT

Qualitative feedback from the software development team suggests that requirements backlog items developed from the MBCD functional models were readily useable in the software development space, and generally required a reduced effort in the sprint planning. This may suggest that the focus on the functional model as a core element of the MBCD derived solution system results in a natural synergy with the functional focus of software applications. It was also noted that provision of the functional models together with their resultant requirements, provided additional clarity to the development team and helped to reduce misinterpretation of the system functional requirements.

MBCD was capable of providing the structure and completeness required to meet the project's assurance and performance requirements, as well as the flexibility required to support evolving stakeholder feedback resulting from the use of Agile. The inherent traceability and completeness of the capability model, allowed for rapid and informed assessment of change impacts. Through use of the model, potential impacts of a change on system assurance and performance aspects could be readily identified.

In certain cases the level of definition within the model was not optimal for the software development team. While for critical assurance and performance aspects of the system a detailed specification was appropriate, many aspects were open to a variety of possible solution implementations. In some cases, detailed functional models and their resultant requirements predicted a specific implementation, unnecessarily constraining the system and preventing an adaptive, agile development that could readily build on stakeholder feedback. System modelling should provide sufficient architecture definition, functional definition and specification to meet assurance and performance requirements while allowing for a flexible and adaptive development approach. Where possible, the system model should define system functions and requirements such that agility in downstream development is maximised. Where flexibility is constrained, active involvement of the software development team in the MBCD process would assist in preventing disconnects between predicted and actual implementation.

FURTHER DEVELOPMENT AND FUTURE APPLICATIONS

Process enhancements

Figure 7 provides a high-level outline of the project workflow employed in the conceptual design and development of the range safety planning tool. The project utilised a requirements set in the form of a system specification to provide a simple linkage between the MBCD work and the software development work. As indicated by the resource allocation bars, the MBCD team developed the conceptual design and then maintained a level of systems engineering support throughout the project. The software development team's involvement in the MBCD phase was limited to specification review, and mostly contained within the final MBCD iteration.

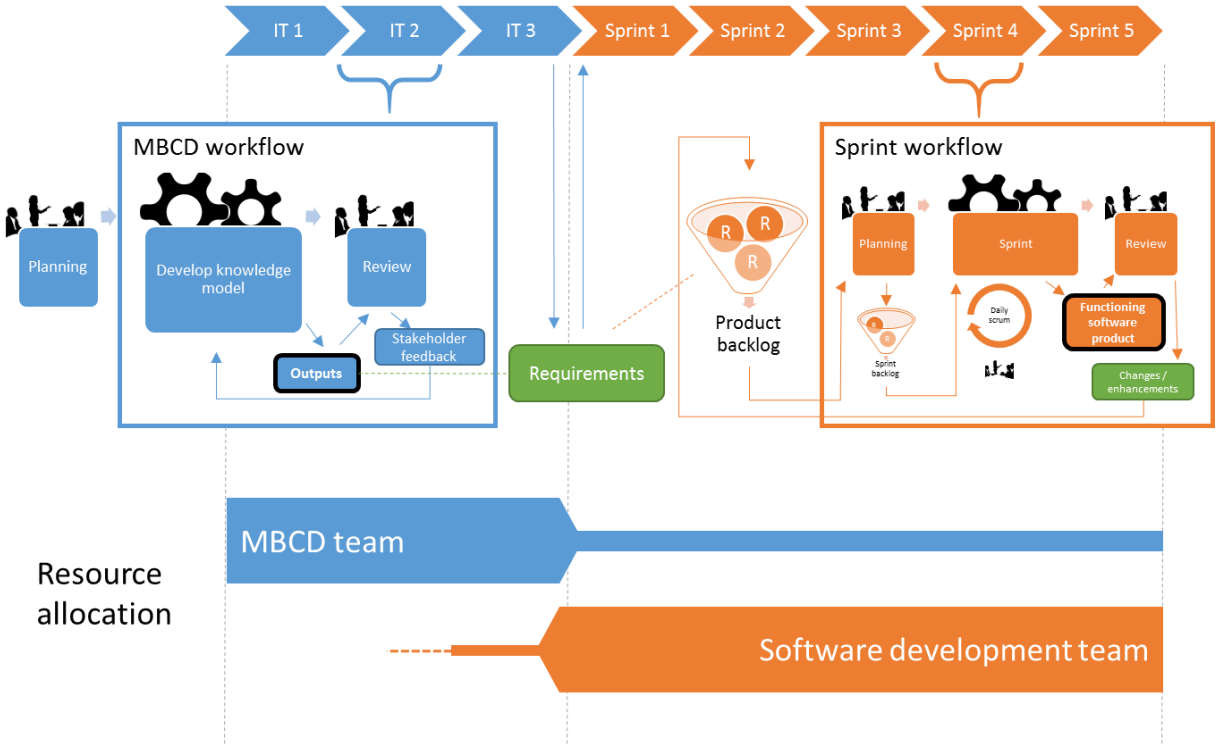

Figure 7. Existing process

Post project review and analysis revealed that improvements might be realised through earlier and increased involvement of the software development team in the MBCD phase (refer resource allocation bars in Figure 8). We propose that software team involvement may be of particular value in definition of software centric aspects of the solution-class definition, and importantly the function modelling of software reliant systems. The software team would be well placed to identify where systems engineers are unnecessarily constraining the software solution, and provide advice on specifying those elements where imposed constraints, and performance and assurance requirements place restrictions on implementation. Furthermore, the software developers could ensure that software elements, such as architectures and software components, were adequately and correctly defined within the model.

A negative aspect of this approach would be the increased resource cost resulting from earlier software development team engagement, however this could be at least partially offset through corresponding reductions in the MBCD team size.

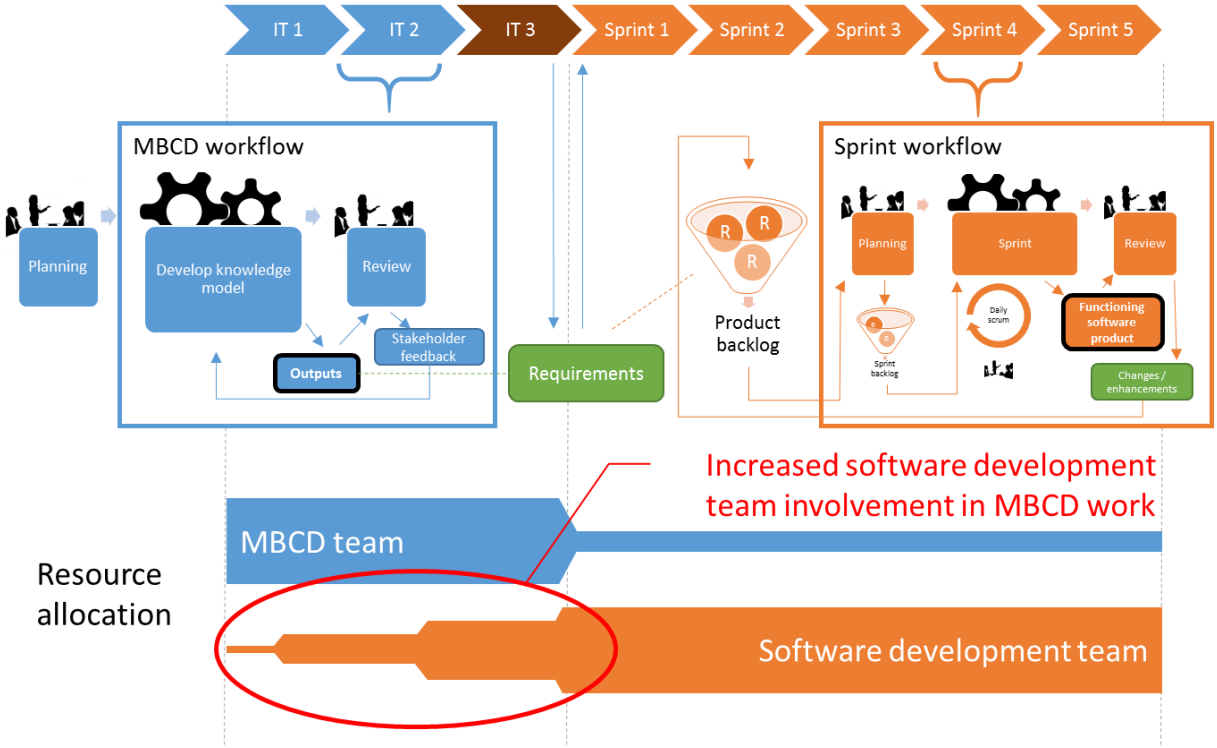

Figure 8. Enhanced process

While our existing iterative spiral development process employed on MBCD projects possesses similarities to an Agile Scrum process, there is potential to change or tailor our MBCD process to align with the Scrum framework and the software development workflow. This alignment of processes is represented in Figure 9. Potential benefits may include better alignment of project management processes throughout both phases, early customer exposure to an agile development environment and mindset, and stricter adherence to the Sprint Planning, Daily Scrum, Sprint Review and Sprint Retrospective events (prescribed by the Scrum framework (Schwaber and Sutherland 2013)) during the MBCD phase.

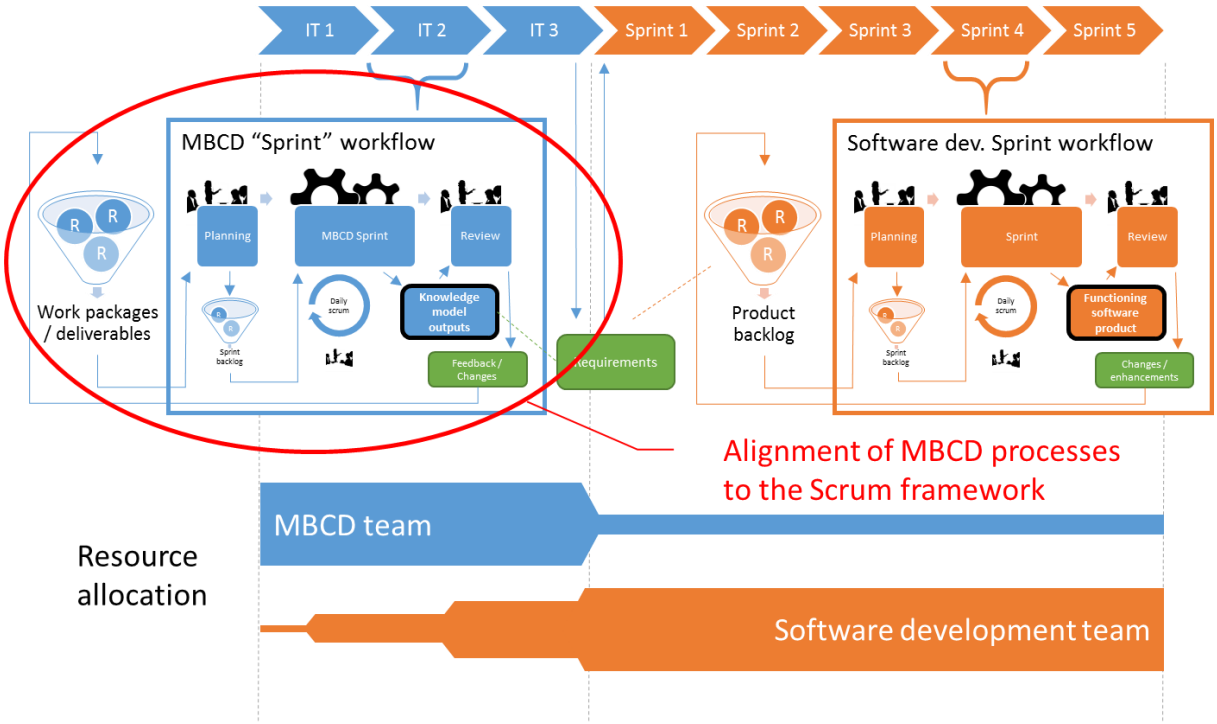

Figure 9. An agile approach to MBCD

An integrated MBCD and software development approach as depicted in Figure 10 might provide further benefits through better resource utilisation and reduced development time. Such an approach would remove the distinct transition between the MBCD and software development phases, providing early involvement of the software development team and allowing earlier kick-off of software development tasks. Such a scenario would allow early software products to influence the MBCD feedback cycle and solution-class definition. The integrated Scrum Team and overlapping MBCD – development approach may help to further prevent misalignment between solution concept and solution implementation.

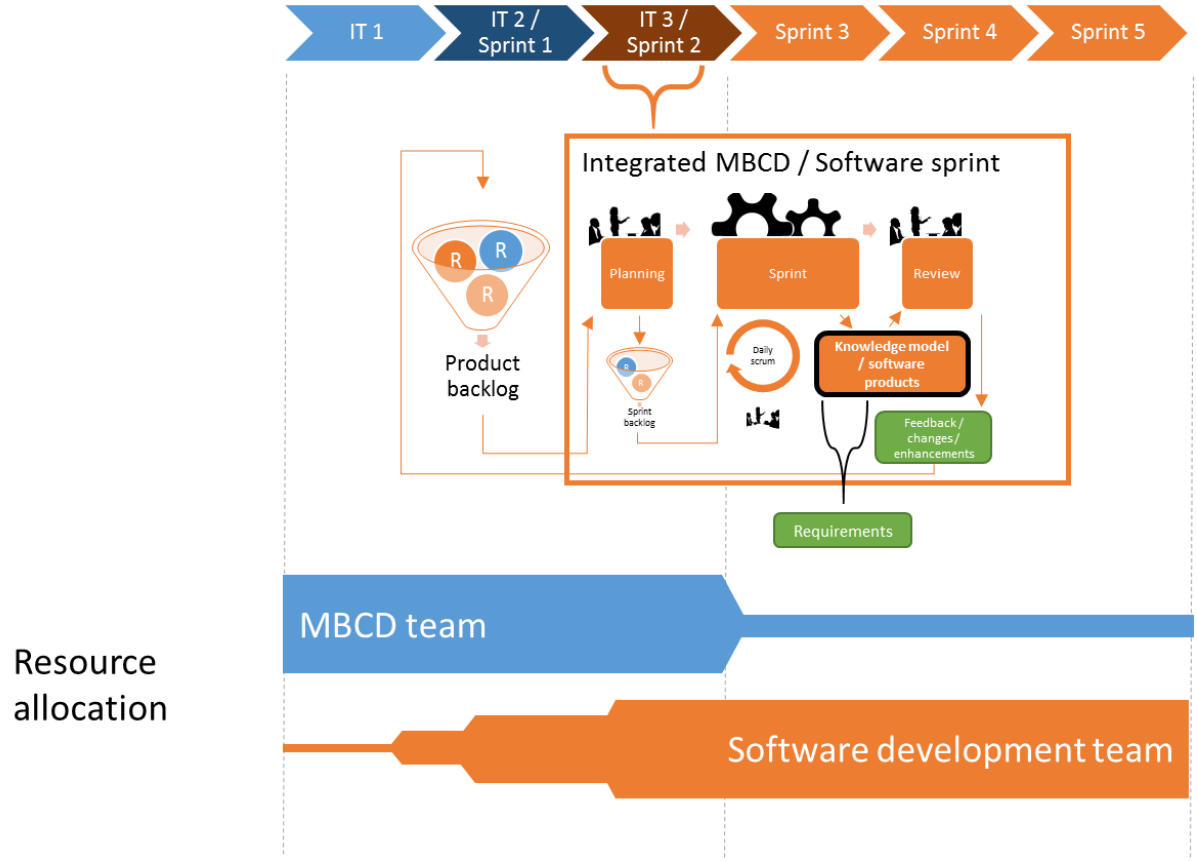

Figure 10. Fully integrated agile development

The process shown in Figure 10 also depicts a scenario where the requirements set (or specification) is no longer the single technical information link between the concept and development stage. Exploring a more complex information exchange between the MBCD model and the software development environment may increase agility by linking functions, architectures and components. Examples might include relating MBCD functional model elements to agile user stories and linking MBCD modelled components to related software components.

Scaling

Having been developed for the definition of large complex Defence capabilities, the WSAF and Shoal's MBCD methods have been proven in delivering high quality large scale capability definition (Robinson et al. 2010 and Tramoundanis & Jones 2012). In addition, techniques such as Scrum of Scrums (Agile Alliance 2013) can be used to scale Agile Scrum for complex software intensive systems (Sutherland 2001). It therefore follows that the integrated approach discussed in this paper should also be scalable. (Rosser et al 2014) provides an Agile Systems Engineering Framework that can be applied to large scale projects. We propose that this framework would be compatible with our MBCD approach.

CONCLUSION

MBCD can be utilised to provide a high quality, defensible and traceable concept definition and system specification. Agile Scrum software development can then be used to realise the MBCD defined system while maintaining stringent performance and assurance levels. The richness of the MBCD model provides the required framework for quickly and accurately assessing the impact of changes on performance and assurance characteristics. Lessons learnt from the case study demonstrated that a careful balance between modelling depth and solution agility was essential to allow developers to adapt the product without unnecessary constraints. Our program retrospective also revealed several potential process enhancements whereby the Scrum framework and agile focus might increase the adaptability and flexibility of the existing layered development approach to MBCD. We also identified that increased developer involvement in the conceptual design, a fully integrated MBCD and development team, and tool interface enhancements may benefit future projects. We propose that given the existing evidence that MBCD and Agile Scrum can be scaled up for complex capability systems, the integration of the processes is likewise scalable.

REFERENCES

Agile Alliance. "Agile manifesto" Online at http://www.agilemanifesto.org, 2001.

Agile Alliance. "Guide to Agile processes: Scrum Of Scrums" Online at http://guide.agilealliance.org/guide/scrumofscrums.html, 2013.

Boehm, Barry, et al. "Architected Agile Solutions for Software-Reliant Systems." *Agile Software Development*. Springer Berlin Heidelberg, 2010. 165-184.

Boehm, Barry, and Richard Turner. *Balancing agility and discipline: A guide for the perplexed.* Addison-Wesley Professional, 2003.

Department of the Army. (2014). *Pamphlet 385–63 - Safety - Range Safety.* Washington DC, USA: Department of the Army. Retrieved from Army Publishing Directorate: http://www.apd.army.mil/pdffiles/p385_63.pdf

Logan, P. and Harvey, D., "Architecting the problem space: an architectural framework-based method for definition of user requirements", Asia Pacific Council on Systems Engineering conference (APCOSEC 2010), Keelung, Taiwan, October 2010.

Robinson, K. and Graham, D., "An improved methodology for analysis of complex capability", Systems Engineering & Test and Evaluation (SETE 2010), Adelaide, Australia, April 2010.

Robinson, K., Tramoundanis, D., Harvey, D., Jones, M. and Wilson, W., "Demonstrating MBSE for specifying complex capability", Systems Engineering & Test and Evaluation (SETE 2010), Adelaide, Australia, April 2010.

Schwaber, Ken, and Jeff Sutherland. "The definitive guide to Scrum: The rules of the game." Retrieved from Scrum. Org: http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-US.pdf, 2013.

Sutherland, Jeff "Agile Can Scale: Inventing and Reinventing Scrum in Five Companies," Cutter IT Journal, vol. 14, pp. 5-11, 2001.

Tramoundanis, D. and Jones, M., "The leap from requirements to capability options", Systems Engineering & Test and Evaluation (SETE 2012), Brisbane, Australia, April 2012.

BIOGRAPHIES

Matthew Wylie

Matthew is a professional engineer with a wide range of systems engineering and project management experience. Matthew has led systems engineering projects and teams in the Defence, automotive and electronics industries; and has experience managing systems engineering projects in all phases of the systems lifecycle. Matthew is a Senior Systems Engineer within the Shoal Engineering MBSE program.

Dr David Harvey

David 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 is currently the Chief Systems Engineer at Shoal Engineering. Shoal has developed an MBSE approach and tailored tool to assist in complex system definition in conjunction with Australian Defence partners. As well as leading this development, he is also involved in applying the tool and approach to capability definition in major Australian Defence projects. David is also the chair of INCOSE's Model-Based Conceptual Design Working Group.

Tommie Liddy

Tommie Liddy is a mechatronic engineer completing his Ph.D. in Robotics at the University of Adelaide while working as a senior Systems Engineer and Programs Manager at Shoal Engineering. 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 Shoal Tommie is developing MBSE tools for operational analysis and capability definition.