Model-based Technical Planning: An evaluation of description techniques

Tomas Huldt, CSEP
Syntell AB
Värtavägen 22, PO Box 10022, SE-100 55,
Stockholm, Sweden
huldt@syntell.se

Jonas Andersson, CSEP
Swedish National Defence College
Drottning Kristinas väg 37, PO Box 27805,
SE-115 93, Stockholm, Sweden
jonasa@fhs.se

Abstract. As model-based systems engineering (MBSE) gains ground as a real alternative for complex industrial projects compared to the traditional plan or document-driven systems engineering paradigm, several challenges remain to be dealt with. MBSE requires not only the technological system-of-interest to be included in the common model repository used for MBSE, but also other elements and structures that are used to describe vital aspects of a complex technical project. This paper elaborates on how the primary planning document, the Systems Engineering Plan (SEP) can be evolved from the document-driven to the model-driven paradigm, and how some SysML diagram types along with architecture framework viewpoints may be used to model the most common content of a SEP. The evaluation is based on industrial experiences using MBSE in real-world industrial development and procurement projects. Several examples on how SysML may be used to represent SEP elements are presented along with suggestions on potential directions for further work.

1 Introduction

In the statues for INCOSE's CSEP systems engineering certification program, *technical planning* is defined as:

"identify program objectives and technical development strategy; prepare Systems Engineering Management Plans, program Work Breakdown Structures, product Breakdown Structures, Integrated Master Plans, and Integrated Master Schedules; identify program metrics including product technical performance measures and key performance parameters, identify program resource needs in terms of equipment, facilities, and personnel capabilities."

Commonly, the technical planning of a project including systems engineering is documented in a written artefact – the Systems Engineering Management Plan (SEMP), by INCOSE referred to as the Systems Engineering Plan (SEP) (INCOSE, 2007). In the domain of systems engineering, a solid SEP describing the overall technological effort in a project, to augment the Project Management Plan (PMP), is regarded instrumental to integrate for instance planning, risk and opportunity management, and specialty engineering areas. Several standards and handbooks gives advise on the structure, content, and purpose of a "good" SEP (IEEE, 1998; INCOSE, 2007; NASA, 1995, 2007).

The trend towards the use of graphical modelling techniques replacing textual representations of e.g. requirements and design documents, commonly addressed as Model-Based Systems Engineering (MBSE), is further leveraged by the recent standardization of SysML as a modelling language for systems engineering (OMG, 2007). MBSE is expected to provide several advantages over traditional document-driven systems engineering promoting more effective collaborative development of complex systems (Friedenthal, 2008).

Based on UML, SysML as a modelling language is well adapted to express structures of elements related to the technological system-of-interest (Andersson, 2000; Herzog, 2005), usually to support a creative development process of a design team. Although useful not only for software intensive systems, less emphasis has so far been spent on its capabilities to express elements needed for e.g. *specialty engineering* areas (e.g. *system safety* and *logistics engineering*), *tradeoffs* between various specialty engineering domains, long term *product data management*, and *technical planning*.

This paper focuses on the aspect of *technical planning* in MBSE, by addressing the question "What will the model-based SEP look like?" In an attempt to contribute to the MSBE development and debate, the paper provides an early evaluation of SysML diagram types for their strengths and weaknesses to provide means to include elements of the SEP in a common model repository of the system-of-interest.

The paper begins with an introduction to technical planning and gives a general introduction to the SEP document. Thereafter, prevalent strengths and weaknesses of the MBSE approach are discussed in the context of technical planning as defined by INCOSE. The paper continues with an evaluation of SysML diagram types as to their capabilities to express common content of a SEP. Finally, the paper ends with conclusions and further work. Throughout this paper, the combination of ISO/IEC 15288 (ISO/IEC, 2002, 2008) and INCOSE Handbook version 3.1 (INCOSE, 2007) which in turn is based on (ISO/IEC, 2002) is used as a normative framework for systems engineering.

2 Technical planning and the need for, and purpose of, a SEP

Considering the concept of planning in general, a good plan should in a consistent way explain how well-defined objectives can be met. It should further provide a realistic view how this can be achieved and provide a roadmap to follow for the work ahead. Careful preparation of a comprehensive plan will on one hand not guarantee success, but lack of a sound plan will on the other hand almost certainly ensure failure. To convey the outcome of the planning process, a planning artefact is commonly prepared. The planning artefact should be as comprehensible as possible in order to promote stakeholders awareness of at least its most vital topics. Planning helps in forecasting the future in terms of the information at hand when the planning activity was conducted. Tools for technical planning include frameworks such as handbooks, templates, and standards to ensure that nothing important is forgotten. A plan should be recurrently revised during the effort planned (Forsberg, 2005; NASA, 1995, 2007).

In the document-driven development paradigm, planning documents, or plans, are used to capture e.g. *vision*, *baselines*, *organization*, and various types of *guidelines* for the project at stake. In complex technical projects, a vital appendix to the PMP is the SEP. The project planning artefact, i.e. the PMP, and the technical planning artefact, i.e. the SEP, are normally separated to reflect the fact that different processes guides project management and systems engineering (ISO/IEC, 2008). One of several reasons for this is to maximize the reusability of processes and methods in an organization, e.g. over the system lifecycle(s) or in different types of projects. Nevertheless, the project and the technical planning processes have some common areas that must be dealt with in order to integrate for instance planning and follow-up of *work tasks*, and *risk management*.

Over years, the SEP has evolved into an artefact in the shape of a document, well-adapted for the document-driven paradigm of systems engineering with its strengths and weaknesses. Several standards and handbooks give advice on the structure, content, and purpose of a "good" SEP (IEEE, 1998; INCOSE, 2007; ISO/IEC, 2007; NASA, 1995, 2007). There are many ways to organize and structure a SEP and its sub-ordinate appendices. In fact,

organizations and projects generally encourage the tailorization of processes and other planning artefacts to serve their purposes while minimizing unnecessary activities for the system-of-interest at stake. Most handbooks and standards suggest some common elements in the SEP. In this paper we adhere to the structure proposed by (INCOSE, 2007), which in turn is harmonized with the requirements of (ISO/IEC, 2008). Below, this content have been grouped based on the structure described in (DoD, 1974):

Technical program planning and control includes a clear description of the target objects for the planning effort, such as the system-of-interest, the organization, and the processes along with their intended use, provided benefits, and artefacts. These should also be traced to measurable objectives of the project, sufficient to determine if the project is on track or not. Furthermore, risks and opportunities should be addressed. A predominant purpose of systems engineering management is the identification and mitigation of technical risks and opportunities. The SEP should therefore include or reference technical risks (and opportunities) along with the process to manage them.

Systems engineering processes, which includes specific tailoring of the systems engineering processes, implementation procedures, trade study methodologies, tools, and models to be used. Examples of areas dealt with are requirements definition and management together with system design. A good basis for establishing systems engineering processes is provided by (INCOSE, 2007) and (ISO/IEC, 2008)

Specialty Engineering Integration describes the integration of specialty areas related to a system-of-interest and balances these to meet technical targets on time and within allocated funding. As these areas may represent diverse engineering domains such as *system safety*, *human factors*, *logistics* and *capability engineering*, the principles to integrate the areas should be included in the SEP together with guidelines how to identity, manage, and mitigate tradeoffs between them.

Temporal constraints (i.e. the Systems Engineering Master Schedule, SEMS, and the Systems Engineering Detailed Schedule, SEDS) should be included together with principles on how to minimize the critical path of tasks in the project, e.g. by a careful selection of development processes in various levels of the system-of-interest.

This list does not claim to be a complete description of SEP content, but rather reflects themes found to be suitable candidates for model-driven technical planning. SEP elements exemplified in this paper include the *scope and vision*, *lifecycle model*, *master schedules*, *technical reviews*, *organization*, and *process definition*.

3 State of the art and practice in MBSE

MBSE is undoubtedly an area that has attracted a large interest during recent years. Using model representations in systems engineering is however not a recent development. Systems engineering as a discipline has relied on graphical models for decades including notations as *Functional Flow Block Diagram* (FFBD), *Data Flow Diagram*, *N-Squared charts*, *Integration DEFinition for Function modelling* (IDEF) Diagrams, and *Pugh* and *Quality function deployment matrices* (INCOSE, 2007). Also, many modelling techniques related to single specialty engineering areas exist. But the prospect of bringing different model representations together in a common model repository together with the recent standardization of SysML and the availability of inexpensive and effective modelling tools, provide a real opportunity for MBSE to become an effective tool also in complex projects.

However, several drawbacks and challenges remain to be dealt with. MBSE only supports engineering of issues effectively expressed in the common model repository used in a systems

engineering process. Issues not expressed by the modelling techniques used in model repository tend to be overlooked, time consuming to deal with, or handled as stovepipes in the systems engineering process. This is emphasized by the shortcomings of SysML (i.e. UML) to represent certain phenomena, hence making them hard to include in the model repository. SysML is fairly strong in describing functional and physical structures, but lack explicit support for e.g. several non-functional system characteristics (Andersson, 2000; Friedenthal, 2008). The use of SysML may also exclude traditional systems engineering modelling techniques such as *QFD* and *N2* charts. Moreover, graphical models are not always more effective and expressive than natural language as they may prove just as hard to interpret unambiguously. *"If you can read a word in more than one way, how many ways can you interpret a picture?"* For instance, graphical models as the basis for contractual agreements constitute an area in which much further work remains. Therefore, the authors of this paper strongly advocate a combination of the model and document-driven paradigms of systems engineering.

However, a potential advantage of using MBSE is its higher precision regarding traceability between elements in the common model repository compared to the prevalent document-driven paradigm, i.e. any element in the model repository may be referenced compared to only pointing out paragraphs, line numbers, or (not uncommon) the document as a whole in the document-driven paradigm. Another drawback of document representations of the system-of-interest is that referencing between documents in most cases is unidirectional and thus easily turns change management into a nightmare, for instance regarding upholding of consistency among documents. The use of structured modelling languages may improve this situation, providing means to put all necessary elements of the system-of-interest and the project in a common data repository, and among other primitives use omni-directional links to represent dependencies and traces between elements in the model repository.

Essential terms in this context are the concepts of a *system-of-interest*, a *model*, and the concept of a *modelling language* (ISO/IEC, 2008; Wikipedia, 2009).

A system-of-interest is a combination of interacting elements organized to achieve one or more stated purposes. As such, a system-of-interest may be considered as a product or as the services it provides. A system element is a member of a set of elements that constitutes a system-of-interest. The elements may represent real or abstract entities.

A model is basically a simplified abstract view of a complex reality therefore partly based on assumptions. This representation may be a physical, mathematical, or logical representation of reality used to describe a system, phenomena, or processes. For development and planning purposes, it may focus on particular views, amplifying the thought process dealing with a complex problem. Models may also be executable allowing them to provide input for simulation that in turn may provide visualization and analysis to gain a deeper understanding of the object of analysis. As the number of assumptions in a model increases, the accuracy and relevance of the model diminishes. In MBSE it is imperative to balance the number of simplifying assumptions and the complexity of the model used.

A modelling language is any artificial language that can be used to express information, or knowledge of a system in a structure that is defined by a consistent set of rules. The rules are used for interpretation of the meaning of the elements in the structure. Examples of modelling languages used in systems and software engineering are the Unified Modeling Language (UML) for software systems, IDEF for processes, and SysML for systems engineering (ISO/IEC, 2008; OMG, 2007). Architectural frameworks such as DoDAF, NAF and MODAF (DoD, 2007; Ministry of Defence, 2008; NATO, 2007) may to some extent provide structure to models of systems and in that sense they may be considered as modelling languages. Although

natural written languages also constitute a model representation of reality it commonly lacks the well-defined formal basis of a modelling language.

4 Evaluation of SysML Diagram types

Modelling languages are limited in terms of what they may describe and which analysis they support. Therefore, the scope of a model and the analysis techniques deployed in the model-driven development of a system-of-interest should be determined early in the systems engineering effort, in order to somewhat lessen the risk of making models that are of no or little use for stakeholders. Sections 5 to 9 discuss the tailoring and use of SysML diagram types, as outlined in Figure 1, to represent some common elements of a traditional SEP document.

The evaluation of SysML diagram in this paper is based on industrial experience from model-driven system development and procurement projects, and a few "close to real" examples of the application of SysML for technical planning are included. The evaluation the SysML has focused on the behaviour and requirements diagram types, whereas structure diagram types are used implicitly to provide a structural context.

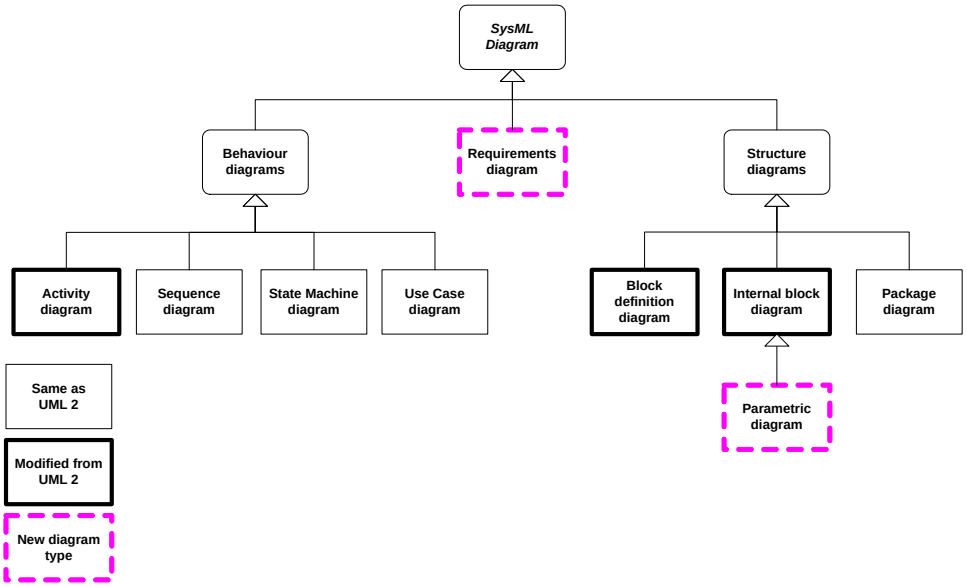

Figure 1: SysML diagram types according to (OMG, 2007).

To provide an idea how these example diagrams presented in the paper "feed" each other during technical planning, and together provide a more complete model representation of the SEP, relationships between the diagrams are outlined in Figure 2.

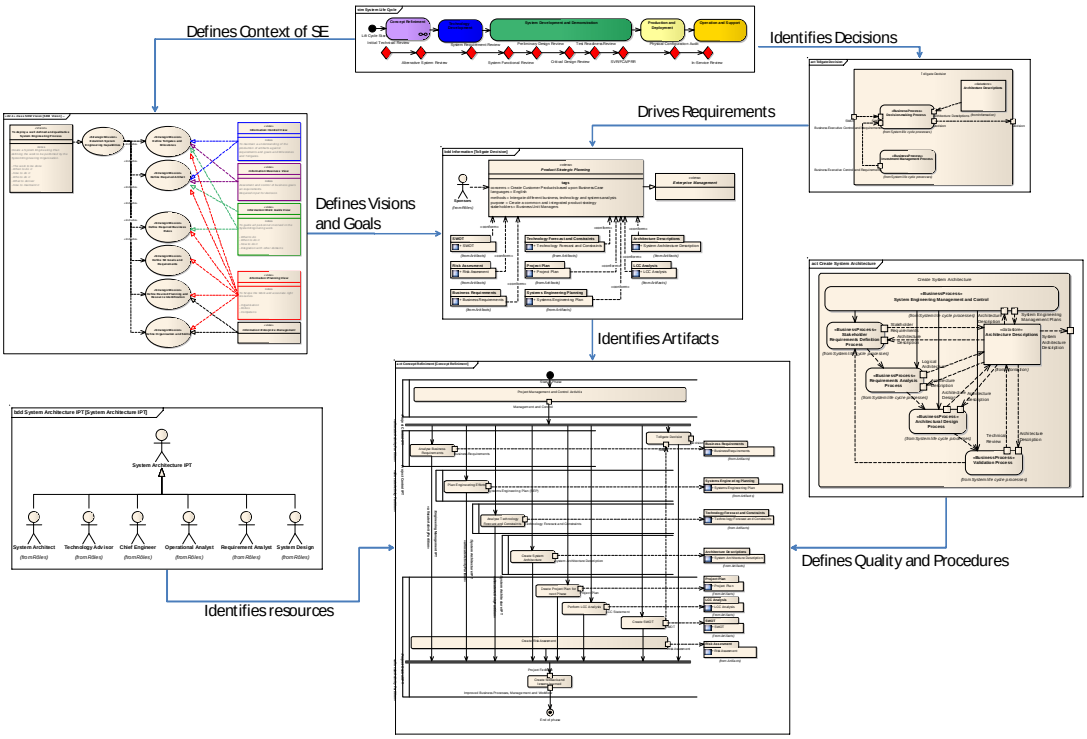

Figure 2. Model relationships.

5 Requirements diagrams

Early in the technical planning, there is a need to capture the purpose and objectives of performing systems engineering, both in terms of its intended effect and its measurable goals. This is required to define the scope and quality of the process itself as well as the artefact produces (INCOSE, 2007). According to the Object Management Group (OMG) (OMG, 2007) a requirement specifies a capability or condition that must (or should) be satisfied. The manner in which SysML defines and manages requirements is according to a traditional hierarchical and text-based requirement management methodology, but is strongly enhanced in terms of traceability and integration to other architectural description domains.

To be compliant with general requirement management methodology and to be capable to create an integrated set of engineering information from *stakeholder requirements* to the definition of *test and verification*, SysML defines a number of *stereotypes*. Firstly, SysML defines *requirements* and *test cases* as stereotyped classes, and secondly SysML defines a set of relationship stereotypes between requirements, and between a requirement and other model objects. These classes and relationships are according to SysML: *requirement* that enables the definition of a text-based stakeholder need. Each requirement object is recognized by its *unique identifier*; *test case* that enables the definition of a verification method which can be used for verification of one or many requirements; *copy* that enables a customer-supplier relationship between requirements and re-use of requirements in different contexts; *derived requirement* that enables a trace between a requirement and its decompositions; *requirement related* that enables the trace between requirements outside the boundaries of its hierarchy breakdown; *satisfy* that enables the trace between a requirement and the object(s) that will satisfy that requirement; *verify* that enables the trace between a requirements and required method of verification; and *nestling* that enables definition of a hierarchy breakdown of

requirements. As the *nestling* is a general mechanism of a class and since a requirement is a stereotype of class, *nestling* is a powerful capability to structure a set of requirements in a strict hierarchy.

Discussion: In general, requirements express stakeholder needs in terms of what to achieve and correlated contexts and constraints. This need can be expressed as e.g. visions, goals, capabilities, processes, functions, performance, characteristics, cost, and time. One major difference between SysML and the architectural framework MODAF is the definition and management of requirements. According to SysML, a requirement is a class of its self and can therefore be instantiated as a unique requirement object, for example the top speed of a vehicle. According to the definition in MODAF there is no explicit requirement class, however requirements are defined as attributes of a class. This implies that each instantiated class (e.g. a system function) has a required level of quality, for example the function *transportation* has attributes such as *acceleration*, *top speed* etc.

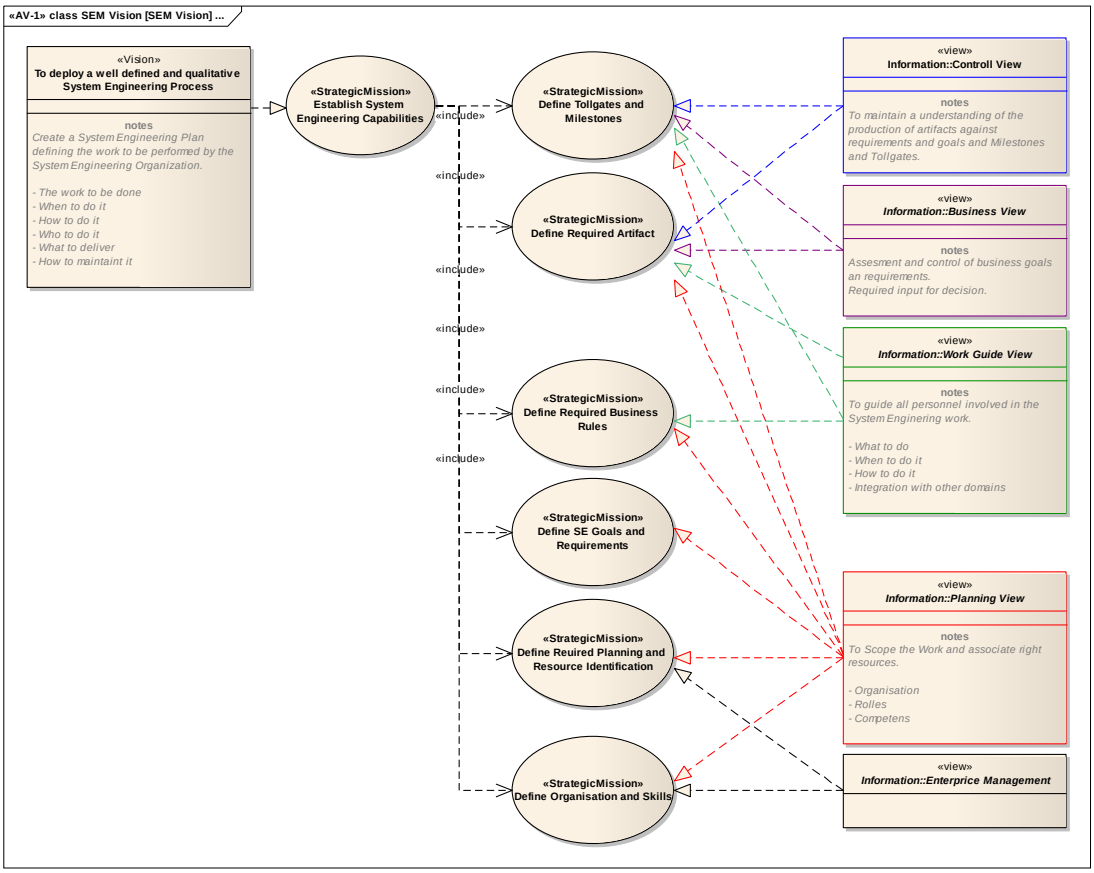

Figure 3. The scope of the Systems engineering plan.

The SysML definition of a requirements class is in accordance with a more traditional systems engineering and requirement engineering methodology, typically resulting in hierarchical structured text-based requirements. These requirements are then the input, constraint, and focal point of all system development efforts and the model is therefore a representation of the logical consequence and solution space of those requirements.

If an architecture framework such as MODAF (Ministry of Defence, 2008) is used, the model itself expresses what is required by a system-of-interest. In other words the model

including all its objects, attributes and relations defines the set of *system requirements* to be realized. As a starting point of the system architecture process, *stakeholder requirements* are analyzed and defined as *system capabilities*. These capabilities identify the context and main purpose of the system-of-interest. When the set of required capabilities are defined, the model is then enhanced with descriptions of how these capabilities are realized throughout a transformation of that need to a system definition. As a consequence, this approach provides a more flexible and rich description of how to express stakeholder needs and concerns (compared with the SysML definition). However text-based requirements are still the most common procedure to express requirements when establishing a contractual acquisition baseline.

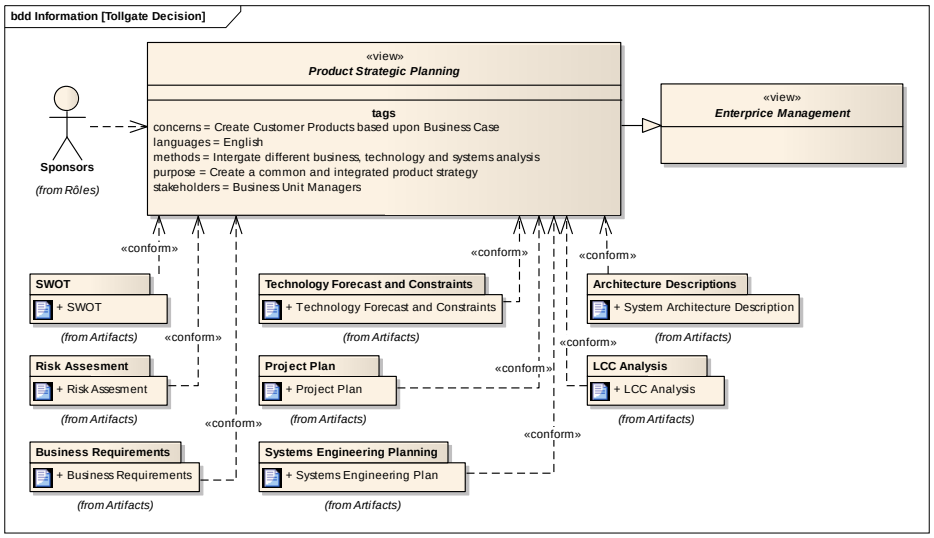

Figure 4. Systems engineering stakeholder requirements.

Example (As described in Figure 3 and Figure 4): When defining the purpose and goal of a model-based SEP a more holistic approach is of use. Therefore it is more applicable to express high level stakeholder requirements as *visions*, *strategic missions*, and *views*. Hence, *visions* define the top requirements of the systems engineering efforts, *strategic missions* define the measurable goals, and *views* define the usage of the systems engineering process output.

Use case diagrams can be used to create traceability between visions and *stakeholder requirements*, *views* and *viewpoints*. By using these diagrams, high level stakeholder concerns can early be captured, traced and validated. This is a vital output when communicating and base-lining the scope and context of planned systems engineering efforts. Refer to Section 6 for more detailed discussion on *use case diagrams*.

Since the model representation of a SEP is stored and maintained in the same data repository as the architecture description of the system-of-interest, it is possible to both define a structure of artefacts during the technical planning process (here modelled as UML packages), and later put the actual output of the systems engineering process in these packages. As a consequence, this allows stakeholders to continuously monitor the progress of the planned systems engineering effort.

6 Use Case Diagrams

Technical planning is responsible for defining the systems engineering organization’s

external interfaces to stakeholders and concerns (INCOSE, 2007). These interactions are often defined as outputs and more explicitly as artefacts supplied by the systems engineering organization.

According to (OMG, 2007), the *use case diagram* is described as:

"The Use-case diagram is a high-level description of functionality that is achieved through interaction among systems or system parts."

Use case diagrams create structures of system behaviours, from a user perspective, in terms of *functions*, *services* and *capabilities*. *Use cases* also create a link between stakeholder needs, in terms of system accomplishments, and associated system functions.

Discussion: Using *use case diagrams* facilitates the expression of systems engineering strategic missions, i.e. what is required by the systems engineering organization in terms of internal or external stakeholders. The *use cases* can be used as the tool to transform the scope of the SEP (defined as a vision) to *stakeholder requirements* (defined as *views*).

Example (As described in Figure 3): In this context, *use case diagrams* are suitable to communicate the overall planning activity (performed by e.g. the systems engineering manager) needed to guide more detailed planning of the systems engineering effort. This combination of *visions*, *strategic missions* and *views* defines the requirements, context and constraints of the SEP as well as facilitating the communication to other domains, internal and external to the systems engineering organisation, such as product management, and research and development.

7 State Machine diagrams

To be able to control the lifecycle of the system-of-interest and its major contractual or business tollgates, the technical planning must define system lifecycle phases (ISO/IEC, 2008) as a part the system-of-interest's lifecycle model.

According to (OMG, 2007), a *state machine diagram* is described as:

"The state machine diagram describes the state transitions and actions that a system or its parts perform in response to events."

State machine diagrams enable modelling of discrete states of system behaviour. The states define isolated and disjunctive set of activities performed during certain circumstances and conditions. The *state machine* also invokes the transition between one state and the next, e.g. condition, rules, logics etc.

Discussion: A SEP should define a system lifecycle model including its *stages*, *tollgates*, and furthermore a set of tailored *processes*, *activities* and *artefacts* for each *stage* (ISO/IEC, 2008). The definition of *states* and *state machines* is therefore a suitable modelling technique for a system lifecycle.

Example (As described in Figure 5 and Figure 6): Each lifecycle stage can be represented by a *state* and a correlated *tollgate decision*. The *state machine* can also be used to describe how different reviews are correlated to the lifecycle. This example use the lifecycle according to The Defence Acquisition University (DAU, 2007).

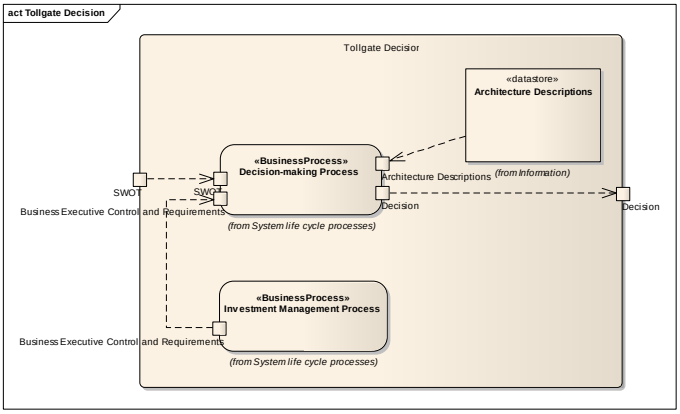

Figure 5. The Tollgate Decision.

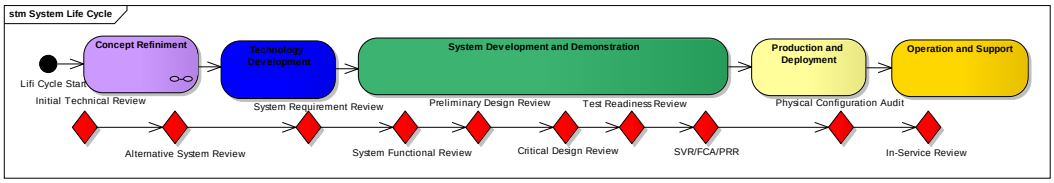

Figure 6. A system lifecycle model.

8 Activity diagrams

To be able to create the base to communicate and allocate tasks and roles to the systems engineering organization, and moreover to be able to perform control of planned tasks, an important outcome of the technical planning is the *Systems engineering detailed schedule.*

According to (OMG, 2007), an *activity diagram* is described as:

"Activity diagram - The activity diagram represents the flow of data and control between activities"

Activity diagrams are the basis for conducting model-driven analysis of activities (e.g. functions, processes, and events). The meta-model in SysML creates a set of rules to describe system behaviour in a solution independent manner, including all aspects of its potential inherent components, software, hardware, personnel etc. SysML also provides an effective toolbox to describe a complex and dynamic reality, *as-is* is or *to-be*. Activity modelling is therefore an efficient method to understand stakeholder needs, system behaviour and inherent requirements and therefore is a cornerstone and prerequisite to system architecture and design.

Discussion: In order to create an overall *Systems engineering master schedule* when developing and operating a system-of-interest, we have introduced the modelling of system lifecycle stages in Section 7. During a lifecycle stage, the system-of-interest is transformed from one *state* to another, i.e. a *state* defines a system condition in terms of *risk level*, *details of design*, *maturity level* etc. At the end of each stage, a decision is made weather to (somewhat simplified): continue to the next stage of the system-of-interest's lifecycle; terminate the project; or iterate back to a previous lifecycle stage. These decisions are made based on knowledge about the system-of-interest often captured in *architectures*, *drawings*, *simulations*, *models* etc. It is often the responsibility of systems engineering to supply stakeholder with engineering data and descriptions sufficient to the current lifecycle stage and its correlated tollgate decision. Technical planning must therefore plan necessary tasks to produce required

outputs.

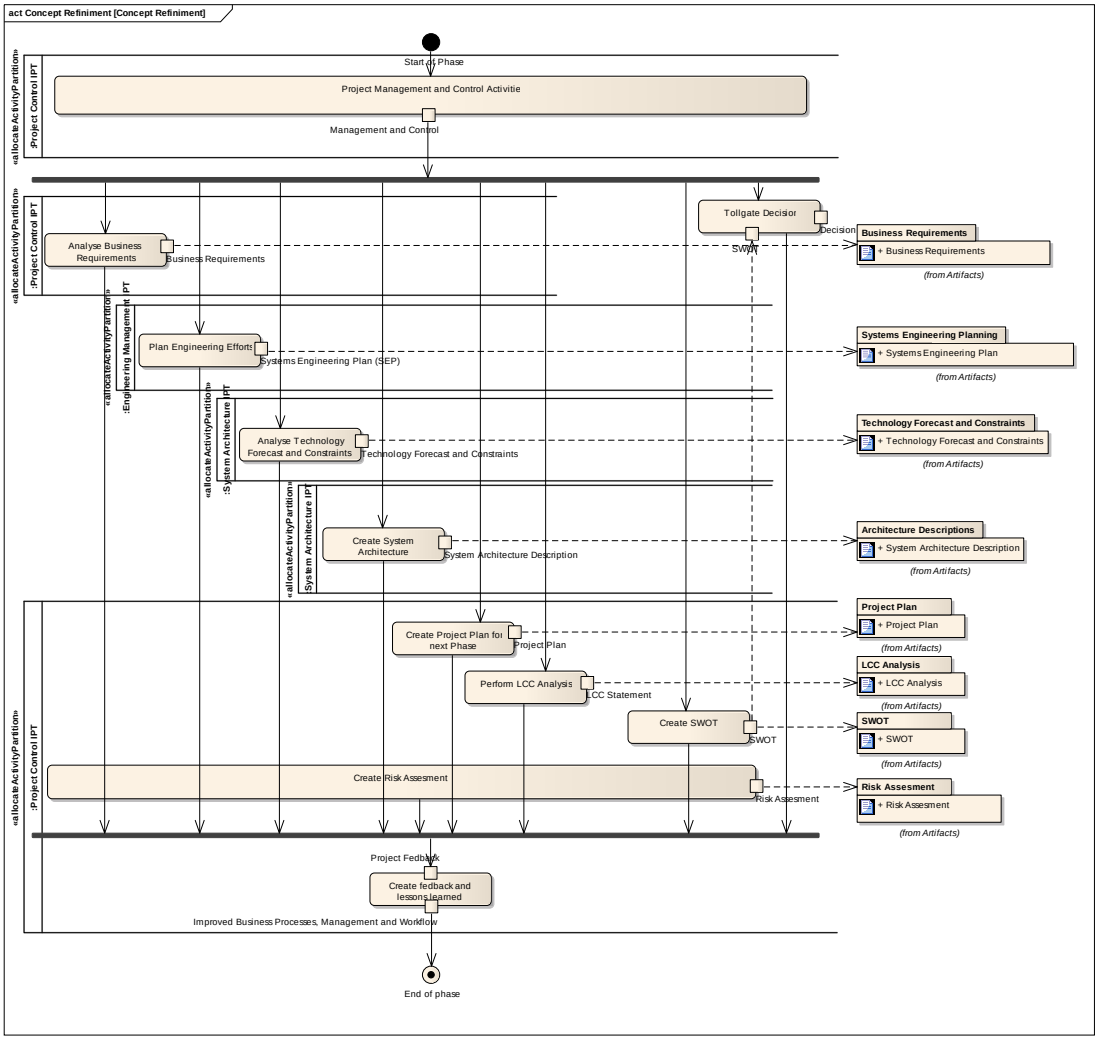

Figure 7. A systems engineering detailed schedule.

Planning example (As described in Figure 4, Figure 7, and Figure 8): In this example the *concept refinements stage* is decomposed using an *activity diagram*. This diagram identifies required tasks to be performed by the systems engineering organization during this stage. Depending on the size and complexity of the system development effort it is probably necessary to introduce several, and decomposed, activity diagrams per stage. To be able to allocate tasks to the organization, *swim lanes* is introduced indicating responsibilities, see Figure 7. Each task has at least one associated artefact as output. These artefacts conform to *stakeholder viewpoints* according to defined *visions*, se Section 5 and 6. If the model is enhanced with *viewpoints* defining each *milestone* and *tollgate*, it is possible to verify the content and usage of all planned and performed tasks according to Figure 4. This methodology is foreseen to make explicit the vital traceability between tasks, output and decisions. Furthermore, this also facilitates a generic capability to trace artefacts to defined *stakeholder requirements* throughout defined *views*, *strategic missions* and finally the *systems engineering vision* according to Figure 3.

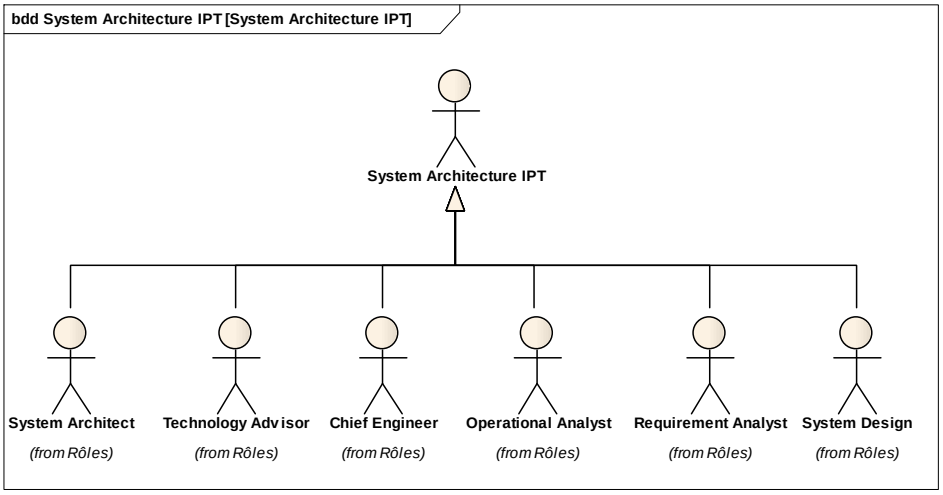

Figure 8. Example of an IPT organisation.

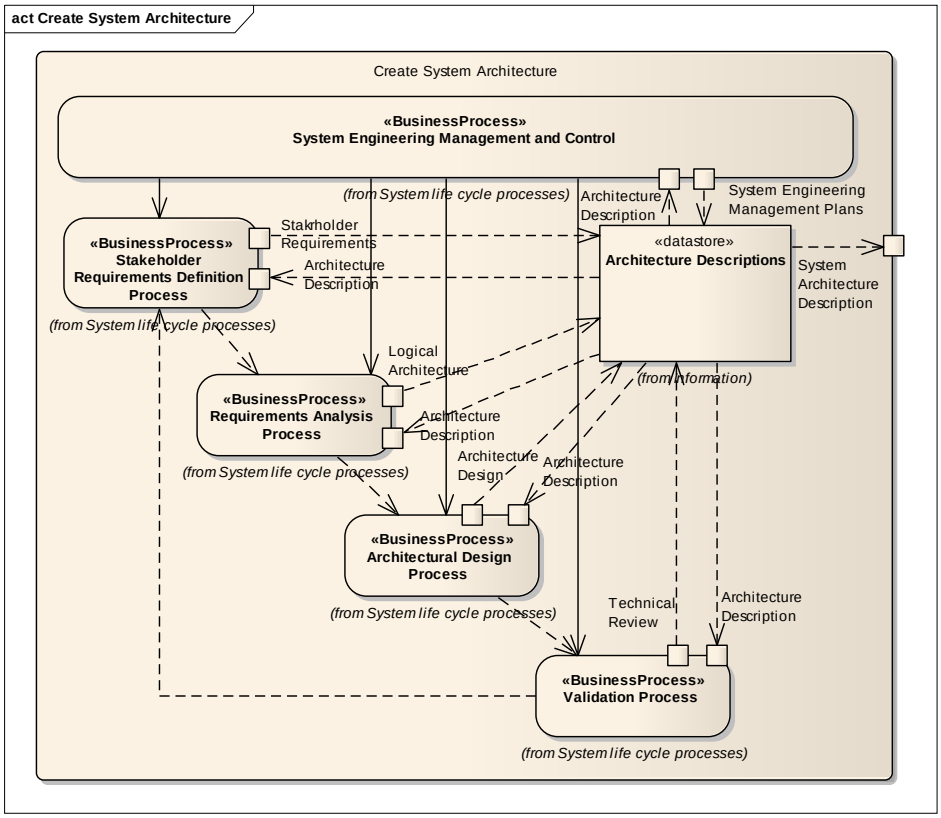

Figure 9. Systems engineering process definitions.

Process definition example (As described in Figure 5 and Figure 9): If the model shall be capable of guiding the work performed by the systems engineering organization, there is a need to introduce *process descriptions* to an applicable level, tailored to the size, skills, and maturity level of the organization, balanced with budgets and time constraints. To make this possible, *activity diagrams* is introduced. In this example, processes identified and defined by (ISO/IEC,

2008) are used as a baseline. To integrate identified tasks with defined processes each task is decomposed to its related *process view*. The *process output* is furthermore identified by packages (see Figure 7) and the process definitions define the content and quality of these outputs.

9 Sequence diagrams

According to (OMG, 2007), the Sequence diagram is described as:

"A sequence diagram represents the interaction between collaborating parts of a system."

Sequence diagrams forms the base to model interactions between actors and/or systems. One of the distinctions between *activity diagrams* and *sequence diagrams* is that the former do not take into account how the actor or system performs its tasks, i.e. how an input is transformed into an output. The *sequence diagram* rather pinpoints the required stimuli and flow between identified actors and systems. A *sequence diagram* is therefore less proficient, compared to *activity diagrams*, to capture the inherent dynamic aspect of system behaviour. However, the *sequence diagram* is an efficient modelling technique to define detailed scenarios, which can be tested and validated. For that reason, *sequence diagrams* are often used in combination with *activity diagrams* to detail the complexity of system behaviour and its interactions.

Discussion: A project or an enterprise organization can be seen as a loosely coupled and dynamic system. This system is realized by personnel and knowledge, processes, methods and infrastructure. As a consequence, tasks may be conducted in many different ways fulfilling the same objectives. *Activity diagrams* is a more powerful method to capture a dynamic behaviour compared to *sequence diagram*, and are therefore more appropriate to model the behaviour of a systems engineering organization.

Example: When modelling dynamical process flows and its allocation to project resources, *activity diagrams* in combination with *swim lanes* have been found more adequate than *sequence diagrams*. Furthermore, *activity diagrams* are more accurate in the context of a SEP as they focus on performed processes, *the how*, rather than the interfaces between system objects (in this case project resources). By addressing *the how*, the model focuses on the quality aspects of performed tasks.

Finally, *activity diagram* is more compliant with traditional *business process engineering* methods and representations, which might facilitate a less complicated deployment in the organization. *Sequence diagrams* is not yet evaluated as an applicable part of the model-based SEP, but is expected to be useful when defining required information infrastructure, in terms of information exchange and workflow procedures. The analysis and implementation of sequence diagrams is one of the logical next steps in introducing MBSE capabilities.

10 Conclusions and further work

Although this paper presents some preliminary experiences of using SysML diagram types to represent the outcome of technical planning, some rather conclusive observations can be made.

Even though SysML is not originally explicitly intended to support technical planning, the language can be successfully applied to represent technical planning elements. This provides a capability in MBSE to represent not only elements of the system-of-interest in a common model repository, but also the planning elements representing e.g. *processes* and their *outputs*, *lifecycle stages*, *work tasks*, and *goals* and *vision* that have important dependencies to the

elements of the system-of-interest. Hence, the overall precision of interdependencies between system and planning elements may be increased dramatically.

However, as more dependencies increases complexity in the model repository representing the system-of-interest, it is found important that the purpose of any model-based technical planning (e.g. stakeholder communication or specific analysis methods) must be established before the start of the systems engineering effort. Also, the analysis methods must be adapted as modelling techniques fit for e.g. functional problem solving may not be the ultimate choice for e.g. lifecycle management of plans or decision support. Consequently, the importance of balancing the number of simplifying assumptions and the complexity of the model used is stressed to mitigate the risk of creating models that are of no or little use to solve the real problems at stake.

A major challenge for MBSE initiatives is the present lack of general guidelines and standards to cover all aspects of systems engineering as described in (INCOSE, 2007; ISO/IEC, 2002, 2008). A common model repository combined with the use of one standardized modelling language is one step in the right direction, but in order to become useful in an industrial-wide scale, there is an urgent need of standardized guidelines and techniques for systems engineering tasks not originally supported by SysML.

Further work. As this paper has presented a first review on how SysML may support model-based technical planning, work still remains to define this as an integrated branch of MBSE. In addition to technical planning, several areas of traditional systems engineering needs to be fully integrated in the MBSE paradigm, such as *specialty engineering* and *tradeoffs between various specialty domains*, *long term product data information management*, *product portfolio management*, and *supportability engineering*. Hence, similar reviews of these areas may be fruitful as they indeed include elements that need to be traced to various elements in the complete model representation of a system-of-interest.

11 References

Andersson, J., Johnson, P. 2000. IT Infrastructure for Electric Utilities: A comparative Analysis of Description Techniques. In *proceedings of the 33rd Hawaii International Conference on Systems Sciences (HICSS-33)*. Hawaii, Maui, USA.

DAU. 2007. DAG, Defense Acquisition Guidebook.

DoD. 1974 Mil-Std 499A - Systems Engineering. Department of Defense.

———. *The DoD Architecture Framework Version 1.5* 2007.

Forsberg, K.M., H.; Cotterman, H. 2005. *Visualizing Project Management - Models and Frameworks for Mastering Complex Systems*. 3rd ed: John Wiley&Sons.

Friedenthal, S. 2008. SysML - Current Challenges and Future Needs. Presentation given at an INCOSE Sweden chapter seminar, November 11.

Herzog, E., Pandikow, A. 2005. SysML – an Assessment. In *15th INCOSE International Symposium*.

IEEE. 1998. IEEE 1220 - standard for application and management of the systems engineering process.

INCOSE. 2007. INCOSE Systems Engineering Handbook v. 3.1. edited by C. Haskins, K. Forsberg and M. Krueger.

ISO/IEC. 2002. ISO/IEC 15288:2002 Systems engineering -- System life cycle processes.

International Organization for Standardization - ISO.

———. 2007. ISO/IEC 42010:2007 Systems and software engineering -- Recommended practice for architectural description of software-intensive systems. International Organization for Standardization - ISO.

———. 2008. ISO/IEC 15288:2008 Systems and software engineering -- System life cycle processes. International Organization for Standardization - ISO.

Ministry of Defence, M. *The MOD Architecture Framework Version 1.2* 2008. Available from http://modaf.org/.

NASA. 1995. NASA Systems Engineering Handbook. edited by R. Shishko.

———. 2007. NASA Systems Engineering Handbook.

NATO. *The NATO Architecture Framework Version 3* 2007.

OMG. 2007. OMG Systems Modeling Language (OMG SysML™), V1.0.

Wikipedia. 2009. Available from http://en.wikipedia.org.

12 Acknowledgements

The authors would like to express their gratitude to Mrs. Ann Allison and Mr. Stuart Allison, Syntell AB, for valuable comments and proof reading. Jonas Andersson would also like to thank the Division of Military-Technology at the Swedish National Defence College for its support of the work resulting in this paper.

13 Biography

Mr. Tomas Huldt is principal consultant in engineering management at Syntell AB in Stockholm, Sweden. He has 14 years experience of consulting and lecturing within logistics and systems engineering, specialized in the acquisition and product definition of defence platforms industrial product portfolios. He is a management team member of Syntell, a Swedish based company that provides expertise to industry and governmental bodies related to development and procurement of complex technical systems. Mr. Huldt holds a M.Sc. in Systems Engineering from the Royal Institute of Technology in Stockholm. He is a member of the Board of directors of the Swedish INCOSE Chapter and is a Certified Systems Engineering Professional (CSEP) and CSEP - Acquisition (CSEP-Acq).

Dr. Jonas Andersson is professor in Military-technology at the Swedish National Defence College (FHS) in Stockholm, Sweden. He has 15 years experience of research, lecturing, and industrial work within Systems and Software Engineering. In parallel with his academic work he is a management team member of Syntell, a Swedish based company that provides expertise to industry and governmental bodies related to development and procurement of complex technical systems. Dr. Andersson holds a Ph.D. in Industrial control and information systems and a M.Sc. in Electrical Engineering, both from the Royal Institute of Technology in Stockholm. He is a founding member and Past president of the Swedish INCOSE Chapter and a Certified Systems Engineering Professional (CSEP).