Help for Systems of Systems: Using UPDM and RAS for SoS Modeling

Matthew C Hause
Atego
5930 Cornerstone Court West
Suite 250, San Diego, CA 92121
917-514-7581
Matthew.Hause@Atego.com

Abstract. Systems of Systems (SoS) modeling is becoming increasingly important in both civilian and military systems. The Department of Defense (DoD) Defense Acquisition Guidebook, [1] defines a SoS as a "set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities." Organizations are changing their emphasis from "We need a new system" to "We need to achieve a specific outcome." As these outcomes become more complex and the associated systems more complex, the management, modeling and simulation of these SoS becomes equally challenging. Often, the SoS is modeled in all its complexity, often at a single level of abstraction or level of detail. Instead of a "mega-model" approach, a standards-based "model of models" approach is what is necessary. This approach will use the Object Management Group (OMG) Unified Profile for DoDAF and MODAF (UPDM) for modeling enterprise architectures from capabilities to detailed components, and the Reusable Asset Specification (RAS) for defining reusable assets. Combining UPDM and RAS provides a Model of Models approach with the main model specifying assets in various levels of detail. The models specified by these assets can be referenced when detailed analysis is required, or hidden when a SoS viewpoint is required, allowing the analyst to see the forest through the trees. The paper will also include an assessment of the applicability and effectiveness of this approach. The International Conference on Systems Engineering (INCOSE) System of Systems Working Group (SoSWG) has collected a set of "Pain Points on SoS" from a variety of international sources. This paper will review these pain points and discuss how the application of a Model of Model approach, combined with standards-based modeling tools and a reusable assets approach can help to alleviate some of the pain currently being felt by SoS architects and managers. Hopefully, this response to their SOS will deliver some much-needed assistance.

Introduction

Systems of Systems (SoS) modeling is becoming increasingly important in both civilian and military systems. Guidance is being written, studies undertaken, standards created and national and international research projects funded. These projects are far-ranging and too numerous to mention, so only a few notable publications and projects will be mentioned. The Department of Defense (DoD) Defense Acquisition Guidebook, [1] defines an SoS as a "set or arrangement of systems that results when independent and useful systems are integrated into a larger system that delivers unique capabilities." The guidebook further emphasizes the importance of systems engineering regarding SoS. "SE is increasingly recognized as key to addressing the evolution of complex systems of systems. SE principles and tools can be used to apply systems thinking and engineering to the enterprise levels. An enterprise in this usage is understood to be the organization or cross-organizational entity supporting a defined business scope and

mission, and includes the interdependent resources (people, organizations, and technology) to coordinate functions and share information in support of a common mission or set of related missions, (reference Federal Enterprise Architecture Framework (FEAF)," September 1999).”[16]
The Systems Engineering Guide for Systems of Systems Version 1.0 August 2008 [2] goes further. It provides “today’s systems engineering practitioners with well grounded, practical guidance on what to expect as they work in today’s increasingly complex systems environment and tackle the challenges of systems of systems. This guide is a step in supporting the systems engineering community to adapt systems engineering processes to address the changing nature of today’s world increasingly characterized by networked systems and systems of systems.” [2]

European Research projects

Ongoing research in SoS includes the COMPASS and DANSE projects. DANSE[4] (Designing for Adaptability and evolutioN in System of systems Engineering), which aims to develop "a new methodology to support evolving, adaptive and iterative System of Systems life-cycle models based on a formal semantics for SoS inter-operations and supported by novel tools for analysis, simulation, and optimization". COMPASS[3] (Comprehensive Modeling for Advanced Systems of Systems), “aims to provide a semantic foundation and open tools framework to allow complex SoSs to be successfully and cost-effectively engineered, using methods and tools that promote the construction and early analysis of models.” COMPASS has provided useful guidance and publications regarding the definition and management of SoS. These include Semi-Formal and Formal Interface Specification for System of Systems Architecture [17], and Model-based requirements engineering for system of systems [18].

INCOSE

The International Conference on Systems Engineering (INCOSE) System of Systems Working Group (SoSWG) has collected a set of “Pain Points on SoS” from a variety of international sources [5]. These will be discussed later in this paper.

Systems of Systems

Systems of Systems are generally large complex systems, with varying degrees of operational independence, managerial independence, evolutionary development, geographical distribution and lifecycle independence. Both individual systems and SoS conform to the accepted definition of a system in that each consists of parts, relationships, and a whole that is greater than the sum of the parts; however, although an SoS is a system, not all systems are SoS. SoS typically are not acquired or specified from scratch. They are modifications to ensembles of existing and new systems which together address capability needs. An SoS can be a logical configuration of existing and new systems, where the systems retain their identity, and management and engineering continue in the systems concurrently with the SoS. Systems can even simultaneously be parts of multiple systems of systems. A family of systems (FoS) is defined as a set of systems that provide similar capabilities through different approaches to achieve similar or complementary effects [6]. FoS are fundamentally different from SoS because, as [6] goes on to say, a family of systems lacks the synergy of a system of systems. However, many of the techniques described in this paper are applicable to FoS.

Systems of Systems Engineering

Systems of systems (SoS) systems engineering (SE) deals with planning, analyzing, organizing, and integrating the capabilities of new and existing systems into a SoS capability greater than the sum of the capabilities of its constituent parts. Consistent with the DoD transformation vision and enabling net-centric operations, SoS may deliver capabilities by combining multiple collaborative and independent-yet-interacting systems. The mix of systems may include existing, partially developed, and yet-to-be-designed independent systems. [1] The emphasis on the capability-driven nature of SoS SE is important to note here. It is the provided capabilities of these systems that provide the criteria to systems engineers to determine how the different systems fit together and whether or not the SoS as a whole will meet stakeholder requirements. Evaluation at the level of individual requirements is too low level, time consuming and complex a manner to determine how to assemble the SoS. This is further emphasized later in this paper when discussing architecture modeling languages. Due to the complexity of these systems, an essential aspect of SoS SE is MBSE.

Model-Based Systems Engineering (MBSE)

Modeling has always been an important part of systems engineering to support functional, performance, and other types of engineering analysis. Wayne Wymore introduced a mathematical foundation for MBSE in his book entitled Model-Based Systems Engineering [8]. However, the growth in computing technology and the introduction of modeling standards such as SysML, UPDM, Modelica, HLA, and others, are helping to enable MBSE as a standard practice, and provide a foundation to integrate diverse models needed to fully specify and analyze systems. Standards such as UPDM and SysML were driven by both industry and tool vendors. This ensures that the standards both meet user requirements and are available in a variety of different commercial and free-ware tools. Without this wide tool support, modelling languages and techniques tend to fall into disuse.

The INCOSE SE Vision 2020 [7] defines Model-based systems engineering (MBSE) as "the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases. MBSE is part of a long-term trend toward model-centric approaches adopted by other engineering disciplines, including mechanical, electrical and software. In particular, MBSE is expected to replace the document-centric approach that has been practiced by systems engineers in the past and to influence the future practice of systems engineering by being fully integrated into the definition of systems engineering processes." Applying MBSE is expected to provide significant benefits over the document centric approach by enhancing productivity and quality, reducing risk, and providing improved communications among the system development team. [7]

The concrete and quantifiable benefits of MBSE are currently being realized in industry today. Hause et al, 2009 [19] in "Testing Solutions through SysML/UML" demonstrated a savings of 70% in testing of safety critical rail signaling systems. Steve Saunders [20] in "Does a Model Based Systems Engineering Approach Provide Real Program Savings?" found a 68% reduction in specification defects since MBSE practices were introduced. This was for a project managing a fleet of submarines. Finally, the NDIA Systems Engineering Division M&S Committee Final Report on Model Based Engineering (MBE) [21] details savings on several projects both quantitative and anecdotal and includes useful guidance on adoption of model-based techniques on complex systems.

Systems Of Systems Architecture

Organizations are changing their emphasis from a platform focus to an emphasis on capabilities. In general, this is a change from "We need a new system" to "We need to achieve a specific outcome." As these outcomes become more complex and the associated systems more complex, the management, modeling and simulation of these SoS becomes equally challenging. The Department of Defense Architecture Framework (DoDAF) [9] and Ministry of Defence Architecture Framework (MODAF) [10] were developed for modeling enterprise architectures from capabilities to detailed components, and the technical and operational management of these architectures as they evolve over time.

Military Architectural Frameworks such as DoDAF and MODAF define a standard way to organize an enterprise architecture (EA) or systems architecture into complementary and consistent views. DoDAF V1.0 contained four basic views: the overarching All Views (AV), Operational View (OV), Systems View (SV), and the Technical Standards View (TV/StdV). Each view is aimed at different stakeholders, and it is possible to create cross references between the views. Although they were originally created for military systems, they are commonly used by the private, public and voluntary sectors around the world, to model complex organizations such as humanitarian relief organizations and public services such as FEMA. Their goal is to improve planning, organization, procurement and management of these complex organizations. All major DoD weapons and information technology system procurements are now required to document their enterprise architectures using DoDAF.

MODAF kept compatibility with the core DoDAF viewpoints in order to facilitate interpretation of architectural information with the US military. However, MODAF v1.0 added two new viewpoints. The new elements were the Strategic and Acquisition Viewpoints called the Capability and Project Views in DoDAF 2.0. These were added to better contribute to MOD processes and lifecycles, specifically the analysis of the strategic issues and dependencies across the entire portfolio of available military capabilities within a given time frame. In MODAF v1.2, Service views were added to support the development of Service Orientated Architectures (SOA). These were based on NAF 3.0 and have been included in DoDAF 2.0. In the same way that the existing views are integrated, the new views are as well. For example, the Project views specify when the capabilities defined within the Capability views will become available. Capabilities can be associated with systems that define the subsystems, organizations and people necessary to achieve required capabilities.

UPDM

The Object Management Group (OMG) Unified Profile for DoDAF and MODAF (UPDM) [13] was created by members of INCOSE and the OMG to define a consistent, standardized means to describe DoDAF V 2.02 and MODAF architectures using the Systems Engineering Language (SysML)[11], [12]. The goals of UPDM are to significantly enhance the quality, productivity, and effectiveness associated with enterprise and system of systems architecture modeling, promote architecture model reuse and maintainability, improve tool interoperability and communications between stakeholders, and reduce training impacts due to different tool implementations and semantics.

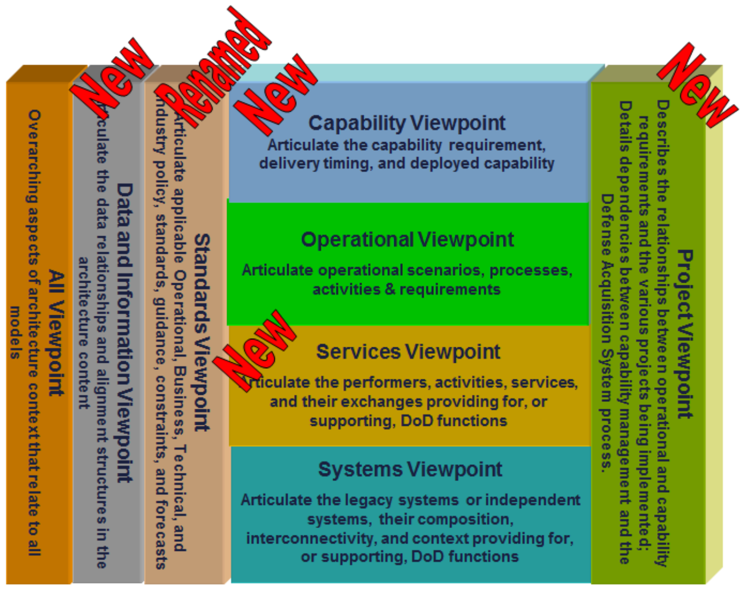

New
Renamed
New
New
New
All Viewpoint
Overarching aspects of architecture context that relate to all models
Data and Information Viewpoint
Articulate the data relationships and alignment structures in the architecture content
Standards Viewpoint
Articulate applicable Operational, Business, Technical, and Industry policy, standards, guidance, constraints, and forecasts
Capability Viewpoint
Articulate the capability requirement, delivery timing, and deployed capability
Operational Viewpoint
Articulate operational scenarios, processes, activities & requirements
Services Viewpoint
Articulate the performers, activities, services, and their exchanges providing for, or supporting, DoD functions
Systems Viewpoint
Articulate the legacy systems or independent systems, their composition, interconnectivity, and context providing for, or supporting, DoD functions
Project Viewpoint
Describes the relationships between operational and capability requirements and the various projects being implemented; Details dependencies between capability management and the Defense Acquisition System process.

Figure 1. UPDM Views (DoDAF 2.0 Version)

The facilities provided by SysML such as physical and logical modeling, interface definition, requirements, functional modeling, parametrics, and multiple levels of abstraction provide powerful modeling capabilities. UPDM is widely used for non-military applications. Use of UPDM will increase substantially as the US Federal Government adopts CAF (a framework that builds on DoDAF) across all federal departments. [14], [22], [24].It is important to stress that UPDM is not a new architecture framework. Instead, it provides a consistent, standardized implementation of DoDAF, MODAF and NAF architectures in UML-based tools as well as a standard for interchange. Figure 1 summarizes the different UPDM viewpoints described above.

The Reusable Asset Specification

Often, the SoS is modeled in all its complexity, often at a single level of abstraction or level of detail. These models consist of complex unreadable, network-style diagrams containing hundreds of objects. These are useless as analysis tools and are a turn-off to non-technical users and managers. Figure 2 provides an example of a typical system architecture diagram. It is a contrived example with the details of the system elements deliberately obscured [23]. The diagram was taken from a blog bemoaning the uselessness of complex system architecture diagrams and is illustrative of many of the diagrams presented by architects to bewildered stakeholders.

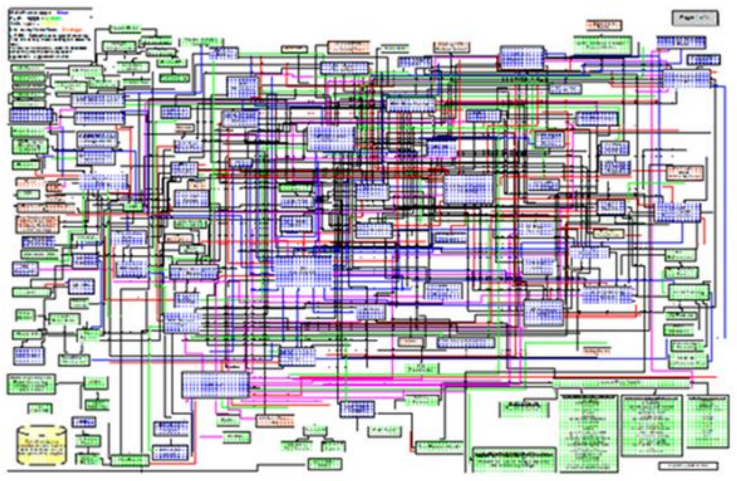


Figure 2 – System of System and Related Interfaces [23]

A Different Approach

Instead of a “mega-model” approach, a standards-based “model of models” approach is what is necessary. The Reusable Asset Specification (RAS) [15] is used for defining reusable assets, their interfaces, characteristics and supporting elements. There are three key dimensions that describe reusable assets: granularity, variability (and visibility), and articulation. The granularity of an asset describes how many particular problems or solution alternatives a packaged asset addresses. The variability and visibility can vary from black-box assets, whose internals cannot be seen and are not modifiable, to white box assets which are visible and modifiable. Two other variations in between are clear-box assets and gray-box assets. The articulation dimension describes the degree of completeness of the artifacts in providing the solution. Asset specifications can also include supporting documentation, requirements addressed, interfaces, etc. Combining UPDM/SysML and RAS provides a Model of Models approach with the main model specifying the system of systems and referencing assets in various levels of detail. The models specified by these assets can be referenced when detailed analysis is required, or hidden when a SoS viewpoint is required, allowing the analyst to see the forest through the trees. To extend the metaphor further, details of the individual trees can also be examined when necessary.

Elements Of SysML

SysML includes diagrams that can be used to specify system requirements, behavior, structure and parametric relationships. These are known as the four pillars of OMG SysML. The system

structure is represented by block definition diagrams and internal block diagrams. A block definition diagram describes the system hierarchy and system/component classifications. The internal block diagram describes the internal structure of a system in terms of its parts, ports, and connectors. The package diagram is used to organize the model.
The behavior diagrams include the use case diagram, activity diagram, sequence diagram and state machine diagram. A use-case diagram provides a high-level description of the system functionality. The activity diagram represents the flow of data and control between activities. A sequence diagram represents the interaction between collaborating parts of a system. The state machine diagram describes the state transitions and actions that a system or its parts performs in response to events.
The requirement diagram captures requirements hierarchies and the derivation, satisfaction, verification and refinement relationships. The relationships provide the capability to relate requirements to one another and to relate requirements to system design models and test cases. The requirement diagram provides a bridge between typical requirements management tools and the system models. The parametric diagram represents constraints on system parameter values such as performance, reliability and mass properties to support engineering analysis. SysML includes an allocation relationship to represent various types of allocation including allocation of functions to components, logical to physical components and software to hardware.

Structural Elements of SysML

The major structural extension in SysML is the «block» which extends the UML Structured Class. It is a general purpose hierarchical structuring mechanism that abstracts away much of the software-specific detail implicit in UML structured classes. Blocks can represent any level of the system hierarchy including the top-level system, a subsystem, or logical or physical component of a system or environment. A SysML block describes a system as a collection of parts and connections between them that enable communication and other forms of interaction. Ports provide access to the internal structure of a block for use when the object is used within the context of a larger structure. SysML provides standard ports which support client-server communication (e.g., required and provided interfaces) and FlowPorts (deprecated in the current version) that define flows in or out of a block. Ports are discussed in more detail below.

Structured Diagram Types.

Two diagrams are used to describe block relationships. The Block Definition Diagram (bdd), similar to a traditional class diagram, is used to describe relationships that exist between blocks. The Internal Block Diagram (ibd) is used to describe block internals. An example of a block definition diagram for a Distiller system is shown in Figure 3. In addition to the system structure, requirements traceability is also shown.

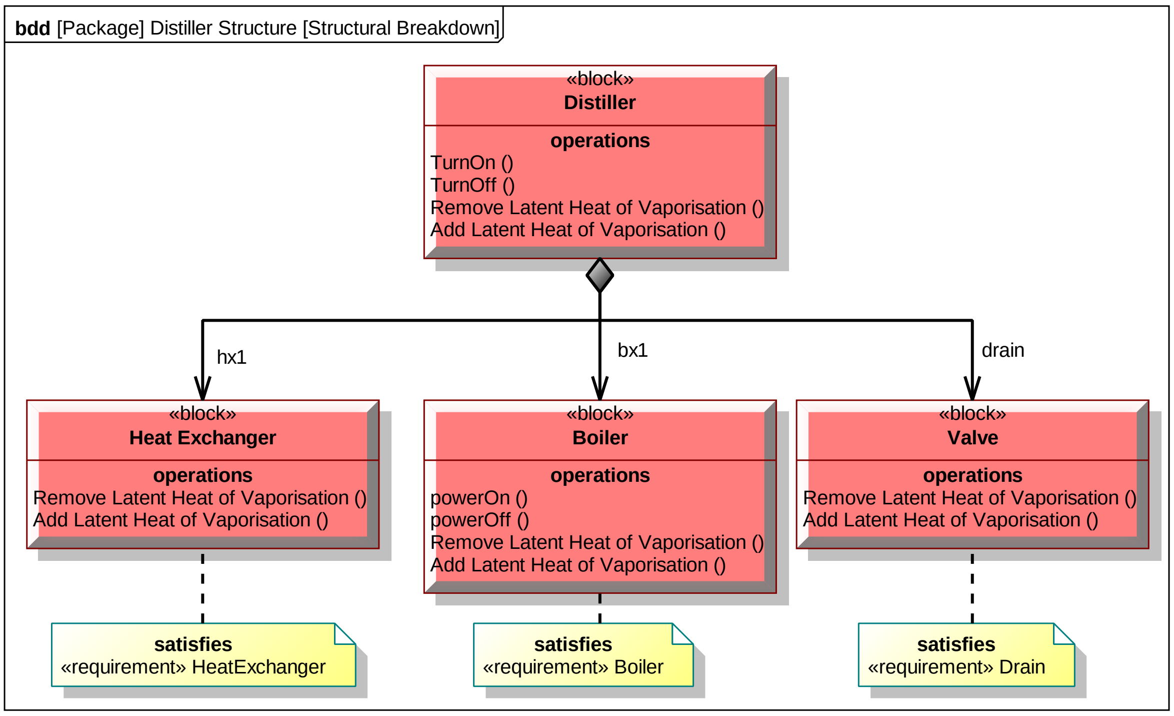

bdd [Package] Distiller Structure [Structural Breakdown]
«block»
Distiller
operations
TurnOn ()
TurnOff ()
Remove Latent Heat of Vaporisation ()
Add Latent Heat of Vaporisation ()
hx1
bx1
drain
«block»
Heat Exchanger
operations
Remove Latent Heat of Vaporisation ()
Add Latent Heat of Vaporisation ()
«block»
Boiler
operations
powerOn ()
powerOff ()
Remove Latent Heat of Vaporisation ()
Add Latent Heat of Vaporisation ()
«block»
Valve
operations
Remove Latent Heat of Vaporisation ()
Add Latent Heat of Vaporisation ()
satisfies
«requirement» HeatExchanger
satisfies
«requirement» Boiler
satisfies
«requirement» Drain

Figure 3 – Block Definition Diagram of Distiller

The Distiller is represented as a block composed of other blocks, including the Boiler, Heat Exchanger and drain Valve. The role names on the association ends correspond to the parts on the ibd. Note the use of compartments to show selected properties of the block; for example, the operations compartment describes the functional behavior of the block. A simple example of an internal block diagram (ibd) is shown in Figure 4.

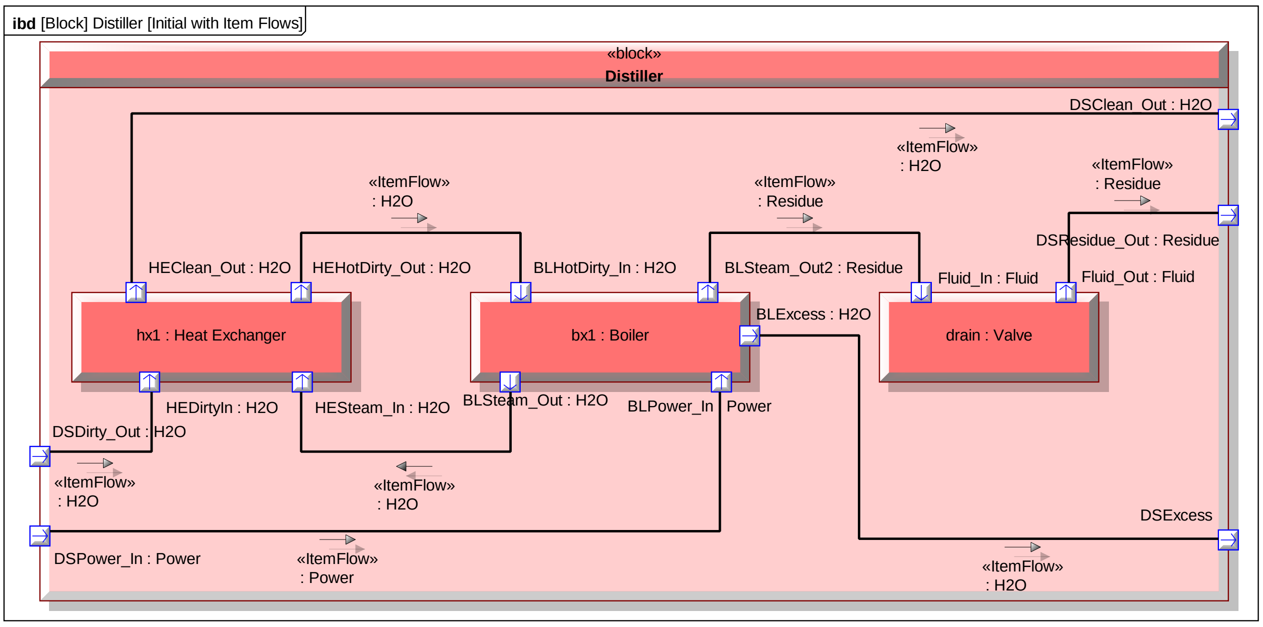


Figure 4 – Internal Block Diagram of Distiller

Figure 4 shows the internal structure of the Distiller block. It shows the hx1, a Heat Exchanger, bx1, a Boiler, and the drain valve that are interconnected to allow material and energy to flow between them and via connections to their parent to external systems. The Heat Exchanger and Boiler each have a number of flow ports that describe what can flow in and out. These are then connected to other compatible ports to enable the required flows in this context. The arrows on the connectors represent item flows that correspond to physical or logical items that actually flow through the system and whose properties can be constrained in parametric models

Standard Ports. Standard Ports are the same as ports in UML 2.0 and used to specify service oriented (request-reply) peer-to-peer interaction which is typical for software component architectures. Standard ports are typed by required/provided interfaces detailing the set of provided/required services. A provided interface specifies a set of operations that a block must provide and a required interface specifies a set of operations that it requires to be provided by another block.

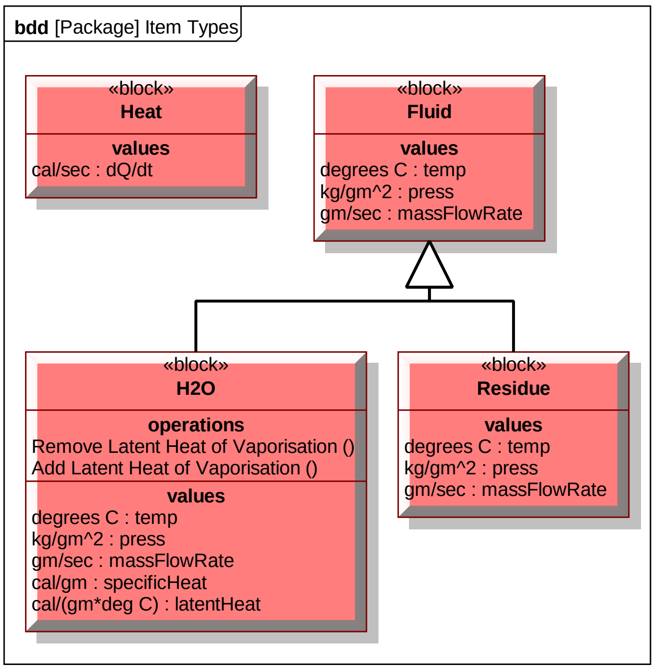


Figure 5 – Block Definition Diagram of Types

Flow Ports. FlowPorts are interaction points through which data, material or energy "can" enter or leave the owning block. A FlowPort specifies the input and output items that may flow between a block and its environment. The specification of what can flow is achieved by typing the FlowPort with a specification of things that flow. This can include typing an atomic flow port with a single item that flows in our out, or typing a non-atomic flow port with a "flowSpecification" which lists multiple items that flow. An atomic FlowPort can be typed by a Block, ValueType, DataType or Signal. Figure 5 shows the blocks used to define the interfaces used in the Distiller system.

Figure 5 shows the types of the interfaces as Heat and Fluid, with H2O and Residue as types of Fluid. The value properties of the blocks are also shown such as temperature, pressure, and mass flow rate. Interfaces can also be defined for data types. Other types of interaction are possible as well. A block representing an automatic transmission in a car could have an atomic flow port that specifies "Torque" as an input and another atomic flow port that specifies "Torque" as an output. A more complex flow port could specify a set of signals and/or properties that flow in and out of the flow port. In general, flow ports are intended to be used for synchronous, broadcast or send and forget interactions. FlowPorts extend UML2.0 ports. Atomic FlowPorts have an arrow inside them indicating the direction of the port with respect to the owning Block. Non-atomic FlowPorts have two open arrow heads facing away from each other (i.e. <>).

Later versions of SysML have further extended ports. Blocks with ports can type other ports (nested ports). Ports nest other ports in the same way that blocks nest other blocks. The type of the port is a block (or one of its specializations) that also has ports. For example, the ports supporting torque flows in the transmission example might have nested ports for physical links to the engine or the driveshaft. [12]

Proxy Ports and Full Ports

SysML identifies two usage patterns for ports, one where ports act as proxies for their owning blocks or its internal parts (proxy ports), and another where ports specify separate elements of the system (full ports). Both are ways of defining the boundary of the owning block as features available through external connectors to ports. Proxy ports define the boundary by specifying which features of the owning block or internal parts are visible through external connectors, while full ports define the boundary with their own features. Proxy ports are always typed by interface blocks, a specialized kind of block that has no behaviors or internal parts. Full ports cannot be behavioral in the UML sense of standing in for the owning object, because they handle features themselves, rather than exposing features of their owners, or internal parts of their owners. Ports that are not specified as proxy or full are simply called "ports." [12] Regardless of how the interfaces have been specified, a consistent and standardized format is required for a viable integration of systems.

Reuse Of Elements In A SoS

As has been stated earlier, SysML blocks are reusable elements. For example, the Valve block can be reused within the architecture within any of the systems as a component. In the Distiller example for instance which is further detailed in the SysML specification [12], an additional valve is added to the design of the distiller. Having defined the valve properties, behavior, interfaces, and constraints, the valve is a known element and does not have to be continually redefined. In addition, if the properties of the Valve block are changed, they are changed for all usages of that block. This is similar to electronic components in a parts catalog. For a simple system such as the valve, there is very little overhead involved in its reuse. When systems start to become complex, this can add considerable overhead to the model in terms of size, performance and complexity. In addition, reuse of components within a model can be problematic. Reuse of model elements within an organization is even more unlikely. In order to achieve effective reuse of design and other high-level assets, along with a re-use culture and good reuse discipline, an organization needs an intelligent asset library that, in effect, goes out and encourages people to find and re-use its assets. The Reusable Asset Specification, coupled with a library of assets and a model of models architecture provides these capabilities. An asset library will also provide a means by which models can be assembled of reused assets rather than built from scratch every time.

To illustrate the problem, we will use an example from a recent INCOSE Tool Vendor Challenge held at the 2013 INCOSE IS in Philadelphia. For further information and the complete solution see http://www.incose.org/symp2013/download/TVC/Atego-TVC-IS2013.pdf . The problem involved an emergency response system. As part of the initial emergency response, ice must be provided to sustain perishables such as medicine and foods, and to support first aid needs. Power and potable water are to be provided with the shelter solution.

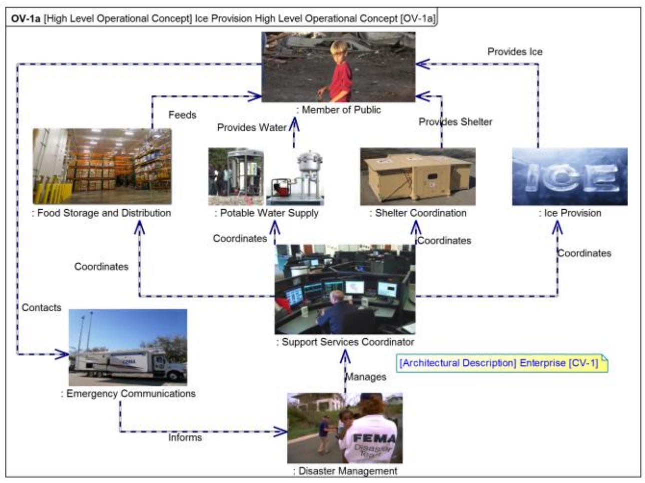

OV-1a [High Level Operational Concept] Ice Provision High Level Operational Concept [OV-1a]
Provides Ice
: Member of Public
Feeds
Provides Water
Provides Shelter
: Food Storage and Distribution
: Potable Water Supply
: Shelter Coordination
: Ice Provision
Coordinates
Coordinates
Coordinates
Coordinates
Contacts
: Support Services Coordinator
[Architectural Description] Enterprise [CV-1]
Manages
: Emergency Communications
Informs
FEMA
: Disaster Management

Figure 6 – Operational Concept for Disaster Relief

The task was to model the system of systems providing the solution and demonstrate how the eventual configuration met the requirements. Subtasks involved the analysis of alternatives, requirements developments, compare and select operational scenario, develop architecture and systems analysis. Figure 6 shows the disaster recoverry concept normally called an OV-1 in UPDM and how ice provision fits into the context.

Capabilities

An ice provision capability is just one of the capabilities required for the disaster response team. This capability also needs to fit in the wider operational context of disaster relief and the lower level capabilities for the provision of food, water and shelter. The UPDM capability views (CV) provide a means of defining required capabilities on the CV-2 diagram. The capabilities can be linked to realizing resources (systems, people, organizations, etc.). The project views provide a means of defining timescales for the deployment of these resources and therefore the realization of the required capabilities. The CV-4 diagram is used to define the dependencies between these capabilities. Figure 7 shows the capabilities required for Disaster Relief and the dependencies between them. The dashed lines with arrows indicate a capability that depends on another. This implies that systems supporting these capabilities depend on the systems that support other capabilities.

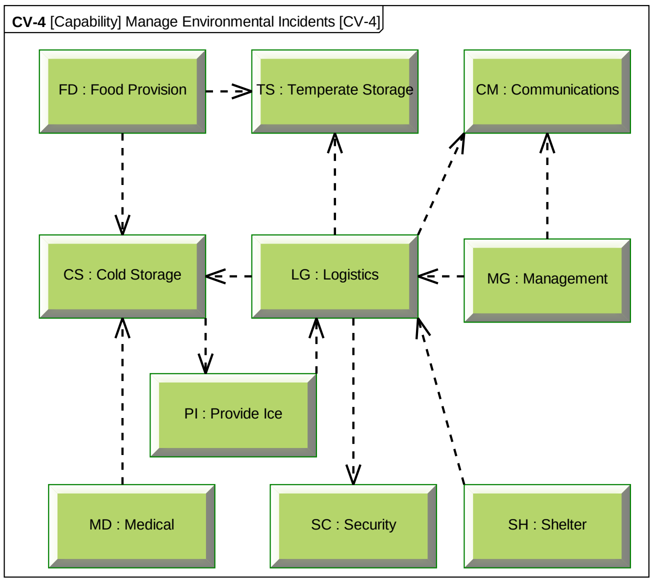


Figure 7 – Disaster Management Capabilities

As shown in Figure 7, Provide Ice requires Logistics for the delivery of the ice. Food Provision requires Cold Storage, which requires ice. And finally, Medical requires Cold Storage, which again requires ice. By mapping out the capabilities as well as the dependencies between them, the architect is provided with a logical system architecture of the eventual system interfaces and dependencies.

Systems Architecture

The Systems views (SVs) describe the physical implementation of the operational and service orientated views and, thereby, define the solution. The SVs are a set of views that describe resources that realize the required capabilities. The SVs describe resource functions and interactions between resources and can also provide detailed system interface models. A recent development in the system views is the involvement of humans in both the operation of systems and in carrying out functions in their own right. This is essential as most systems involve humans as their operations. The SVs can be used to specify solutions to requirements specified in the OVs, or simply to provide more detail to the logical OV architecture.

The Resource Interaction Specification or SV-1 defines how the resources are structured and how they interact and collaborate to realize the capabilities as well as the logical architecture. Multiple variations can be defined to determine an optimal configuration of systems as well as which systems are fit for purpose. The resources can be decomposed to any level required by the architect and depending on the complexity of the solution architecture. Figure 8 shows the SV-1 architecture for victim support.

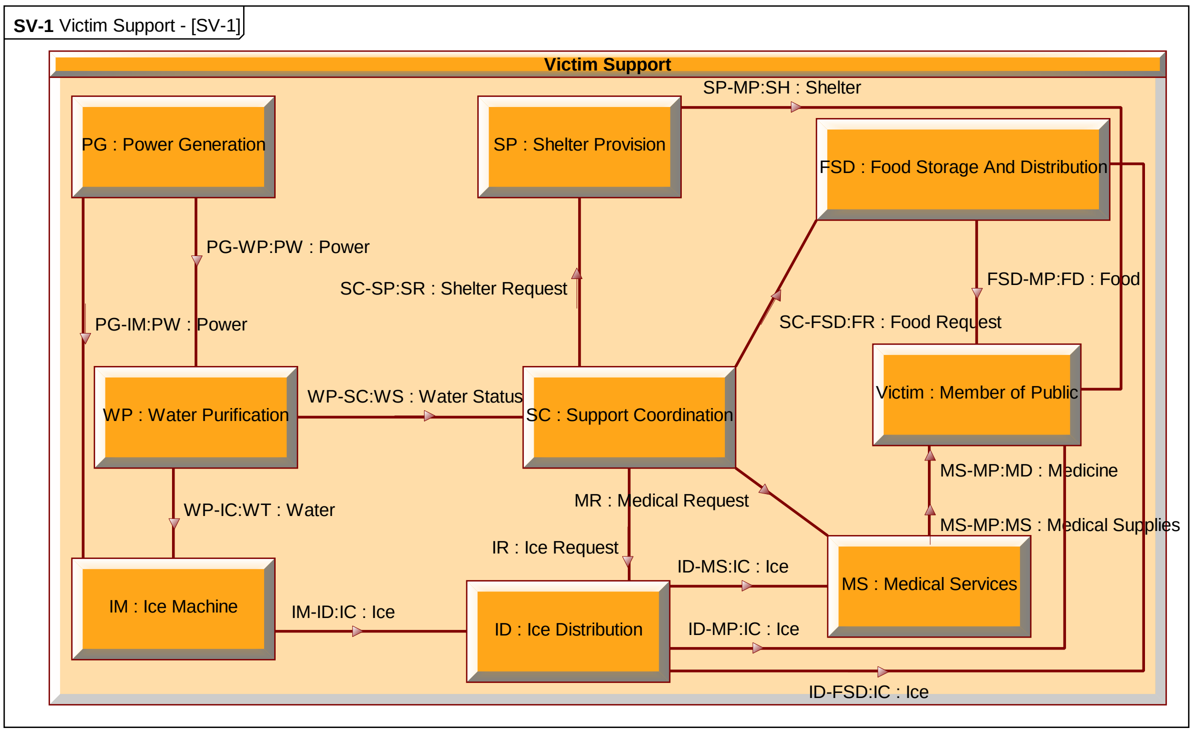


Figure 8 – System Architecture for Victim Support

Figure 8 shows the interactions between the different systems. Interactions include not only data, but also systems, power, and materiel including ice. In the same way that blocks can be reused throughout an architecture, resources can also be reused. Having defined the systems and their interface requirements, more detailed interfaces can be elaborated and specified in detail in the SV-2 diagram. These interfaces are similar to those defined using the SysML Internal Block Diagram (IBD) as shown in Figure 9. Figure 9 shows the Ice Plant configuration with the specific systems that will be deployed. The UPDM implementation of DoDAF uses SysML Blocks and Ports interfaces explicitly. This is because systems and materiel are defined as specialized SysML blocks. Additional systems such as the Fuel Supply had been identified during analysis as the Power Supply will need fuel to operate.

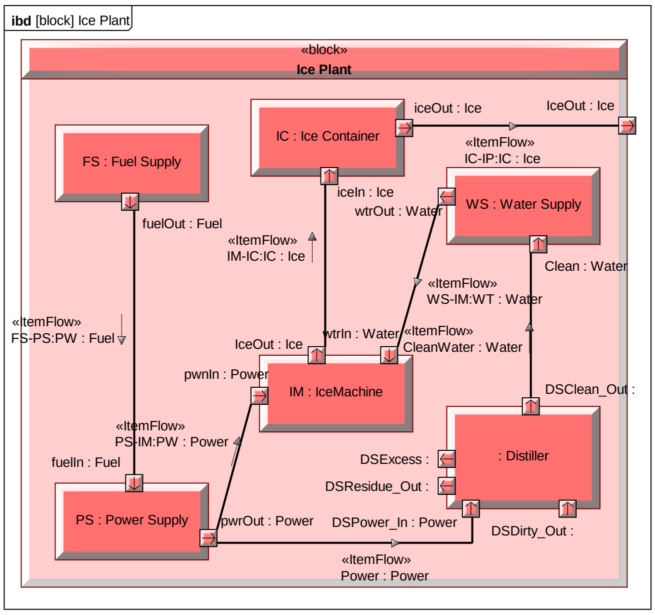
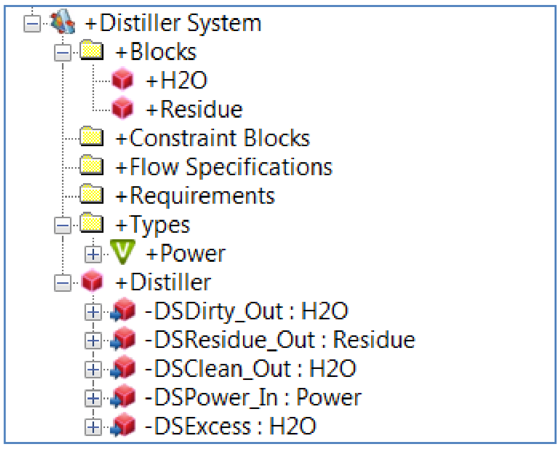


Figure 9 – System Architecture for the Ice Plant

In addition, the Distiller defined previously has been reused in this model. Figure 10 shows the reused Distiller system in the model browser.

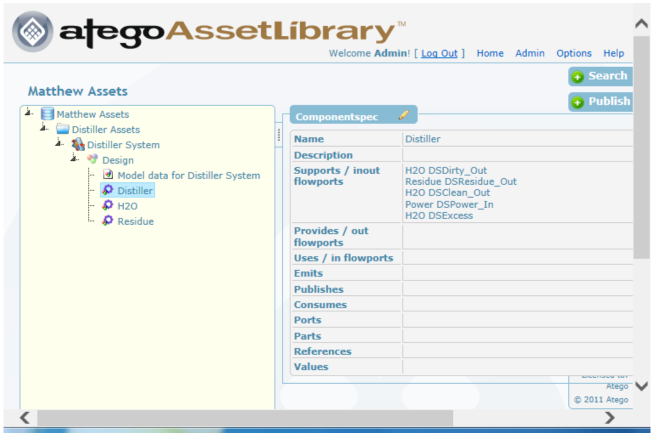
Figure 10 shows the reused characteristics of the distiller system. These include the blocks, constraint blocks used for parametrics, flow specifications, requirements associated with the distiller, types and the interface ports and attributes owned by the distiller. This provides a means of inserting the distiller system into the architecture for reuse. Figure 9 shows the Distiller connected to the Power Supply and Water Supply via the Power and Clean Water ports. The Excess, Residue and Dirty Water ports have not yet been connected. This indicates a need for additional systems and procedures and personnel for the system. This structure mirrors that found in the Distiller asset in the asset library as shown in Figure 11



```
+Distiller System
  +Blocks
    +H2O
    +Residue
  +Constraint Blocks
  +Flow Specifications
  +Requirements
  +Types
    +Power
  +Distiller
    -DSDirty_Out : H2O
    -DSResidue_Out : Residue
    -DSClean_Out : H2O
    -DSPower_In : Power
    -DSExcess : H2O
```

Figure 10 – Distiller System Asset in Model

Figure 11 shows an example of an asset library with the defined Distiller system in an asset catalog. Included with the asset is a description, interface, references, values, etc. This level of detail is appropriate for reuse of the system as a black box component in a system of systems. However, since the requirements, interfaces, behavior, and parametric characteristics are also included, quantitative as well as qualitative analysis can be done on the asset to determine if it is the best fit for the problem at hand. Included with the asset is the specification of the source model from which the asset definition was taken. The architect can access the source model when these details are required, described above as examining an individual tree. Assets stored in the library can be searched, versioned, updated and queried by architects looking to reuse component elements. In addition, engineers can express interest in an asset and be informed when it has been updated. If required, the entire model in all of its detail can also be stored along with the asset. However, this lessens the benefits of the asset library and the reuse of systems. Otherwise, engineers may resort to typical "clone and own" methods of development. In this scenario, the traceability to the original component is lost along with the attendant benefits of reuse.



```
Matthew Assets
  Distiller Assets
    Distiller System
      Design
        Model data for Distiller System
        Distiller
        H2O
        Residue
```

Name	Distiller
Description	
Supports / inout flowports	H2O DSDirty_Out Residue DSResidue_Out H2O DSClean_Out Power DSPower_In H2O DSExcess
Provides / out flowports	
Uses / in flowports	
Emits	
Publishes	
Consumes	
Ports	
Parts	
References	
Values	

Figure 11 – Distiller System Asset in Library

Finally, the RAS interface to the asset library is essential to ensure interoperability of tools and models. There are a variety of modeling tools on the market supporting SysML, UPDM and other methods. Often multiple tools are used within the same project for legacy as well as budgetary reasons. The standards-based definition of the assets provides a means of reusing assets modeled in a different tool without having to support bespoke tool to tool interfaces. The UML XML Metadata Interchange (XMI) interchange format is currently the means by which UML-based tools can exchange data. However, complete interchange has not yet been achieved. See [14] for more information. In addition, the tool independent interface will allow non UML-based tools to reuse assets as well.

Finally, the examples used in this paper were by necessity simple so as to illustrate the techniques and concepts in such a way as they can be easily understood. The problem with all examples is that if they are too simple, then the techniques for more complex problems have not been proven. If they are too complex, then the concepts are too difficult to understand. Therefore, it is worth noting that SysML, UPDM, and RAS have been used for the past several years on projects that were both complex and ultimately successful [19], [20], [21].

System Of System Pain Points

The International Conference on Systems Engineering (INCOSE) System of Systems Working Group (SoSWG) has collected a set of "Pain Points on SoS" from a variety of international sources. These include Lack of SoS Authorities and Funding, leadership, Constituent Systems, Capabilities and Requirements, Autonomy, Interdependencies and Emergence, Testing Validation and Learning, and SoS Principles. [5]. As stated in the abstract, this paper explores the combination of UPDM, RAS, SysML and an asset library in order to determine whether or not they can provide relief the SoS pain points listed previously. The techniques defined previously cannot help with all of these. However, it is worth looking at these briefly to examine whether or not any impact can be made. MBSE has been around for quite some time and only recently has the return on Investment (ROI) been demonstrated via quantitative studies. In the same way, it is hoped that over time, the use of these techniques will demonstrate their ROI and alleviate the pain currently being felt on projects.

Lack of SoS Authorities and Funding

No modelling technique or procedure can provide authority or funding. However, as MBSE has begun to demonstrate true ROI, it is hoped that these techniques can provide ROI to decrease the cost of developing SoS. Built into the asset library is an analysis system that provides metrics in terms of cost savings for reuse. This built-in capability will provide automatic analysis of ROI often lacking in processes.

Constituent Systems

Integrating constituent systems is difficult, especially when these systems are already in existence and often not well understood. Clearly defined system interfaces, capabilities, requirements, behavior, characteristics, etc. is essential for any meaningful integration. By separating out the individual system models, and integrating them as black box systems, engineers can concentrate on the individual system definitions without worrying about the other systems. Also, having clearly defined the interfaces to the systems, development of these systems can take place in parallel without affecting other models and the systems. Integration in the SoS model can then examine the interaction of the individual systems as a whole.

Capabilities and Requirements

As stated above, defining systems in terms of capabilities provides a means of determining the purpose and benefits of a system at a very high level. Capabilities describe desired outcomes as well as specifying stakeholders and realizing resources. By defining systems in this way, architects and engineers can determine capability overlaps as well as capability gaps. These capabilities can then be evaluated using the technique described in Figure 7 to determine how the systems work together at a capability level. This is particularly useful when no model exists of the existing system(s). When models do exist, detailed system functions as well as the requirements that they satisfy can also be a part of the SoS model for more detailed analysis examination.

Autonomy, Interdependencies and Emergence

As spelled out in the report, emergent behavior is often unpredictable. As an experienced systems engineer of many years, I am aware that the real problems of systems integration only come to light when they are being integrated together in the field under real conditions. The modeling and simulation of these systems and SoS can certainly help, but no test or set of tests could ever predict all possible outcomes. Recent problems with the Boeing Dreamliner have certainly borne this out. However, as modeling simulation techniques improve and a critical mass of system models is built up, problems involving emergent behavior can be found, diagnosed and mitigated before the systems are fielded.

Testing Validation and Learning

The models used for simulations and test of the rail system described in Hause, 2009 [19] had been developed over the space of 10 years. They were extremely complex safety critical systems involving the interaction of multiple complex systems. It was only when this critical mass of the system model was reached that the system testing became practical and lead to the extraordinary ROI of 70% savings on systems testing. This is not always possible or even practical and will depend on individual project circumstances. It should also be noted that the engineers involved did not have automated testing in mind when the system models were originally built. It was only through experimentation that these benefits were realized.

Conclusions

Systems of Systems (SoS) have been in existence for some time. However, the definition, analysis and modeling of these SoS is a fairly recent phenomenon. This has been driven by the complex nature of today's systems for commercial reasons, as well as to support government guidance such as net-centric warfare. Modeling these systems has yielded mixed results for a variety of reasons already spelled out in this paper. Rather than simplify the analysis of the SoS, modeling has often further complicated it. This paper has provided examples of how a standards-based approach using UPDM, RAS, SysML and an asset library will allow architects and engineers to assemble complex systems models without the attendant overhead encountered by existing techniques. They will also provide the real possibility of reuse that up until now has eluded both software and systems engineers. As MBSE has started to demonstrate ROI, it is hoped that these techniques will provide ROI in time, money and superior systems architectures and possibly some relief for the pain that we have all been feeling.

References

[1] DoD, 2013, Defense Acquisition Guidebook, Production Date:15 May 2013 Available at http://at.dod.mil/docs/DefenseAcquisitionGuidebook.pdf, Accessed Nov, 2013

[2] DoD, 2008, Systems Engineering Guide for Systems of Systems Version 1.0 August 2008, Available at http://www.acq.osd.mil/se/docs/SE-Guide-for-SoS.pdf, Accessed Nov, 2013

[3] COMPASS, 2013, Available at http://www.compass-research.eu/, Accessed Nov, 2013

[4] DANSE, 2013, Available at https://www.danse-ip.eu/home/index.php/about/main-project-objectives, Accessed Nov, 2013

[5] Dahmann, J, 2013, SoS Pain Points & Implications for MBSE, Presented at INCOSE IW, 2013, Available at http://www.ndia.org/Divisions/Divisions/SystemsEngineering/Documents/NDIA-SE-MS-SoS_2013-08-20_Dahmann.pdf

[6] Chairman of the Joint Chiefs of Staff (CJCS), 2007(1), CJCS Instruction 3170.01F Joint Capabilities Integration and Development System, Washington, DC: Pentagon, May 1.

[7] INCOSE SE Vision 2020, September 2007, Available at http://www.incose.org/ProductsPubs/pdf/SEVision2020_20071003_v2_03.pdf, Accessed Nov 2013

[8] Wymore, A Wayne, 1993, Model-Based Systems Engineering, CRC Press, Boca Raton, Florida, ISBN 0-8493-8012-X

[9] DoD CIO, 2012, DoD Architecture Framework Version 2.02, DoD Deputy Chief Information Officer, Available online at http://dodcio.defense.gov/dodaf20/dodaf20_pes.aspx, accessed Nov, 2013.

[10] MOD Architectural Framework, Version 1.2.004, May, 2010, Office of Public Sector Information, https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/36757/20100602MODAFDownload12004.pdf

[11] Friedenthal, S., Moore, A., Steiner, R. Practical Guide to SysML: The Systems Modeling Language Second Edition, Morgan Kaufman, Oct 31, 2011

[12] Object Management Group (OMG), June, 2012, OMG Systems Modeling Language (OMG SysML™), V1.3, OMG Document Number: formal/2012-06-01, http://www.omg.org/spec/SysML/1.3/PDF/, Accessed November, 2013

[13] UPDM, 2013, Object Management Group (OMG), 2013, Unified Profile for DoDAF/MODAF (UPDM) 2.1, formal/2013-08-04, available at http://www.omg.org/spec/UPDM/2.1/PDF

[14] Hause, M, 2013, Rebuilding the Tower of Babel - The Case for a Unified Architecture Framework, presented at INCOSE International Symposium, 2013 proceedings

[15] OMG, 2005, Reusable Asset Specification (RAS), Version 2.2, http://www.omg.org/spec/RAS/2.2/PDF, formal/05-11-02

[16] Federal Enterprise Architecture Framework (FEAF), Version 1.1, September 1999, Available online at http://www.enterprise-architecture.info/Images/Documents/Federal%20EA%20Framework.pdf, Accessed Nov, 2013

[17] Jeremy Bryans, Richard Payne, Jon Holt, Simon Perry. Semi-Formal and Formal Interface Specification for System of Systems Architecture. In Proceedings of the 7th International Systems Conference, IEEE SysCon 2013, IEEE, April 2013.

[18] Jon Holt. Model-based requirements engineering for system of systems. In Proceedings of the 7th International Conference on System of System Engineering, IEEE SoSE 2012, IEEE, July 2012.

[19] Hause, M. Richards, D. Stuart, A., , June, 2009, Testing Solutions through SysML/UML, INCOSE IS 2009

[20] Steve Saunders, FIEAust CPEng, Raytheon, Does a Model Based Systems Engineering Approach Provide Real Program Savings? Informal Symposium on Model-Based Systems Engineering DSTO, Edinburgh, South Australia, 2012

[21] Jeff Bergenthal (Subcommittee Lead), NDIA Systems Engineering Division, M&S Committee, February 2011, Final Report, Model Based Engineering (MBE), Subcommittee, available online at http://www.ndia.org/Divisions/Divisions/SystemsEngineering/Documents/Committees/M_S%20Committee/Reports/MBE_Final_Report_Document_(2011-04-22)_Marked_Final_Draft.pdf

[22] Okon, Walt, 2012, Moving Towards a Unified Architecture and the US Government Information Sharing Environment, 1105 Enterprise Architecture Conference, 28 November 2012

[23] NQ Logic, If It Ain't Broke, Fix It, Available at http://www.nqlogic.com/2009/07/cit-definition.html, accessed Nov 2013

[24] [WH 2012] The Common Approach To Federal Enterprise Architecture, Executive Office Of The President Of The United States, May 2, 2012, http://www.whitehouse.gov/sites/default/files/omb/assets/egov_docs/common_approach_to_federal_ea.pdf

Biography

Matthew Hause is Atego's Chief Consulting Engineer, the co-chair of the UPDM group a member of the OMG Architecture Board, and a member of the OMG SysML specification team. He has been developing multi-national complex systems for almost 35 years. He started out working in the power systems industry and has been involved in military command and control systems, process control, communications, SCADA, distributed control, office automation and many other areas of technical and real-time systems. His roles have varied from project manager to developer. His role at Atego includes mentoring, sales presentations, standards development, presentations at conferences, specification of the UPDM profile and developing and presenting training courses. He has written over 100 technical papers on architectural modeling, project management, systems engineering, model-based engineering, human factors, safety critical systems development, virtual team management, systems development, and software development with UML, SysML and Architectural Frameworks such as DoDAF and MODAF. He has been a regular presenter at INCOSE, the IEEE, BCS, the IET, the OMG, DoD Enterprise Architecture, Embedded Systems Conference and many other conferences. He was recently a keynote speaker at the Model-based Systems Engineering Symposium at the DSTO in Australia. Matthew studied Electrical Engineering at the University of New Mexico and Computer Science at the University of Houston, Texas. In his spare time he is a church organist, choir director and composer.