The SysML Modelling Language

Fifth European Systems Engineering Conference
18-20 September 2006
Matthew Hause
Artisan Software Tools, Eagle Tower Suite 701, Cheltenham, Glos. UK,
MatthewH@Artisansw.com

On July 6th 2006, the Object Management Group™ (OMG™) announced the adoption of the OMG Systems Modeling Language (OMG SysML™) as a final adopted specification. The OMG SysML specification was in response to the joint Request for Proposal issued by the OMG and INCOSE (the International Council on Systems Engineering) for a customized version of UML 2 designed to address the specific needs of system engineers. The OMG SysML specification was developed by a broad-based team including tool vendors, leading industry users, government agencies and professional organizations over a period of 3 years. Creating the SysML specification, for which Artisan was the Specification Architect has been a mammoth task in which over 100 man-years of effort has been invested. OMG SysML is a visual modelling language that extends UML 2 in order to support the specification, analysis, design, verification and validation of complex systems that include components for hardware, software, data, personnel, procedures and facilities. OMG SysML is intended to be used with different methodologies including structured analysis, object orientation and others. OMG SysML reuses a subset of UML 2 concepts and diagrams and augments them with some new diagrams and constructs appropriate for systems modelling. This paper will look at the background of OMG SysML and summarize the OMG SysML specification including the modifications to UML 2.0, along with the new requirement and parametric diagrams.

INTRODUCTION

The Unified Modelling language (UML) has, since its adoption in 1997, proved immensely popular with software engineers to the point where it is now the only widely used visual modelling language for software engineering. In the past, UML's software focus has discouraged many system engineers from adopting it in earnest. There were various approaches used to address the shortcomings of UML for Systems Engineers. Some made use of the stereotypes provided in UML to create "libraries" or profiles of entities in their application domain. By applying these, they were able to express non-software concepts (Hause, 2003). Others used tools such as Microsoft Visio to model their Systems Engineering concepts, in conjunction with their UML model. However, with this approach they were left with two separate models that they were unable to integrate or cross-reference. Some simply ignored the problem or used words to fill the gap. Some tool manufacturers such as Artisan Software Tools extended UML, allowing integration of Hardware, Software, and Systems Engineering concepts in a single model (Hause 2001). This left them open to a charge of being "non-standard". However, as most Systems Engineers are pragmatists, this argument was not usually effective. In fact, most of the concepts outlined in Hause (2001) have been integrated into UML 2.0 and OMG SysML. Lykins (2000) provides an early paper on trying to adapt UML to SE and some of the issues.

Shortcomings of UML. Those who know UML find it to be an effective modelling language. The roots of UML are firmly in software OMG (1997) states that the "Unified Modelling Language (UML) is a general-purpose visual modelling language that is designed to specify, visualize, construct and document the artefacts of a "*software system*" [italics mine]. In the UML 2.0 version of the specification, *software system* has been replaced by *system* [OMG, 2003a]. This reflects the increased use of UML by systems engineers and users from other domains. The UML is sufficiently flexible and robust to support extensions to address the needs of systems engineering. One of the strengths of UML is its built-in mechanisms for specialising the generic forms of its modelling elements to more application-specific variants. Collectively, these provide a capability for UML "Profiles" that package specific terminology and substructures for a particular application domain. Exploiting this achieves a "standard modelling language for Systems Engineering to analyze, specify, design, and verify complex systems, intended to enhance system quality, improve the ability to exchange Systems Engineering information amongst tools, and help bridge the semantic gap between systems, software, and other engineering disciplines" (OMG SysML, 2003). However, the modifications to UML needed for Systems Engineers require more than just the addition of stereotypes.

Problems with UML 1.x. The starting point for many Systems Engineers is the System Context Diagram which includes a depiction of the input/output flows between the system and/or components, interfaces and the elements in its environment. The only UML 1.x diagram capable of modelling physical nodes is the Deployment

Diagram. Unfortunately, OMG (2003b) states that the Deployment Diagram is used to "define the execution architecture of systems that represent the assignment of software artefacts to nodes." This rather limits its scope. Its most basic fault is the inability to model hierarchical system architectures in any detail. Object Sequence Diagrams (OSDs) were also clumsy to use. There was no way to link to other sub-sequences, such as «include» or «extend» use cases. As much of the Systems Engineer's work in UML dealt with the use case or behavioural view, this became awkward and repetitious. Additionally, there was no way to model or link to requirements, or to model parametric equations.

UML 2.0. UML 2.0 went some way towards addressing the problems of modelling architectures. In particular, it provided enhanced capability for modelling hierarchical structure and behaviour. The Composite Structure Diagram provides a hierarchical architecture; however, it originally only allowed one level of hierarchy and did not allow the modelling of flows on links. Object sequence diagrams were also improved. It is also worth noting that early collaboration between SE and the UML 2.0 teams resulted in some of the issues being addressed. However, links to requirements, parametric equations and others were still not addressed.

Goals of OMG SysML. Consequently, the decision was made by the Object Management Group (OMG) to pursue UML for systems engineering. In March 2003, the OMG issued a Request for Proposal (RfP) for a customized version of UML suitable for Systems Engineering written by the OMG Systems Engineering Domain Special Interest Group (SE DSIG). Friedenthal, Burkhart, (2003) gives early history on the development of the UML for SE RFP. The customization of UML for systems engineering is intended to support modelling of a broad range of systems which may include hardware, software, data, personnel, procedures and facilities. The goal is to provide a "standard modelling language for systems engineering to analyze, specify, design and verify complex systems, intended to enhance systems quality, improve the ability to exchange systems engineering information amongst tools and help bridge the semantic gap between systems, software and other engineering disciplines" (OMG SysML, 2003). There was only one technology submission to the RfP, called OMG SysML. The OMG SysML Team has a broad range of members including system engineers, tool vendors, government organizations and academic institutions. ARTiSAN Software Tools has been an active member of the OMG SysML Team. The OMG SysML logo is shown in Figure 1.

Figure 1: The OMG SysML Logo

OMG SYSML STRUCTURE

This material is based on the draft v1.0 OMG SysML specification which is still subject to minor change throughout the OMG finalization process. The following sections outline some of the additions and modifications made to UML 2.0 to support OMG SysML. It will by no means be a complete list, but will touch on the major highlights for Systems Engineers. Many of the descriptions and examples are taken directly from the specification and an OMG SysML tutorial developed by the OMG SysML Team with additional information added for those readers not familiar with UML. For a complete description the reader is directed to the OMG SysML specification, (OMG, 2006) which can be found on the OMG SysML website (www.omgsysml.org). Unlike many other UML specifications it contains a worked example and is written in language that is understandable to those who are not methodologists.

New Diagrams. OMG SysML reuses a subset of UML 2.0 and provides additional extensions needed to address the requirements in the UML for SE RFP. OMG SysML adds new diagrams to UML and modifies others. One of the goals of the OMG SysML Team was to try and use UML 2.0 as much as possible and avoid making changes unless absolutely necessary. During the initial stages, many new diagrams were considered, however, these have now been cut down to just two, the Requirements Diagram and the Parametric Diagram, both of which are variations on existing UML 2.0 diagrams. The Venn diagram in Figure 2 shows the relationships between UML and OMG SysML.

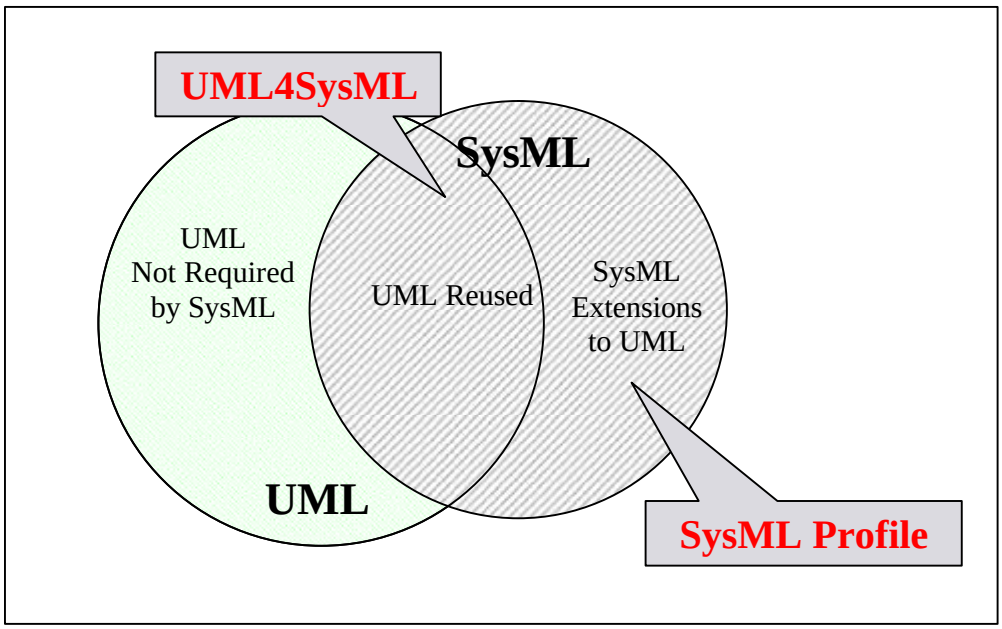

Figure 2: UML/OMG SysML Relationships

The sets of language constructs that comprise the UML and OMG SysML languages are shown as the circles marked "UML 2" and "SysML", respectively. The intersection of the two circles, shown by the region marked "UML Reused," indicates the UML modelling constructs that OMG SysML re-uses. The specification lists the UML packages that an OMG SysML tool must reuse in order to implement OMG SysML. The region marked "SysML Extensions to UML" indicates the new modelling constructs defined for OMG SysML which extend existing UML constructs. Note that there is also a part of UML 2 that is not required to implement OMG SysML, which is shown by the region marked "UML Not Required by SysML".

Compliance. Compliance requirements for tools are also defined in the specification. This is also now in the UML specification (OMG, 2006). Compliance with OMG SysML requires that the subset of UML required for OMG SysML is implemented and the extensions to the UML subset required for OMG SysML are implemented. In order to fully comply with OMG SysML, a tool must implement both the concrete syntax (notation) and abstract syntax (metamodel) for the required UML subset and the OMG SysML extensions. UML has three compliance levels (L1, L2 and L3) that OMG SysML applies to the subset in the UML4SysML package. Compliance to a higher level (e.g., L3) requires compliance to the lower levels. In addition to UML, further units of compliance for OMG SysML are the sub packages of the SysML profile. Tool vendors will be required to state whether their support is full compliance, partial or no compliance for each unit of compliance.

Diagram Summary. The OMG SysML diagram Taxonomy is shown in Figure 3.

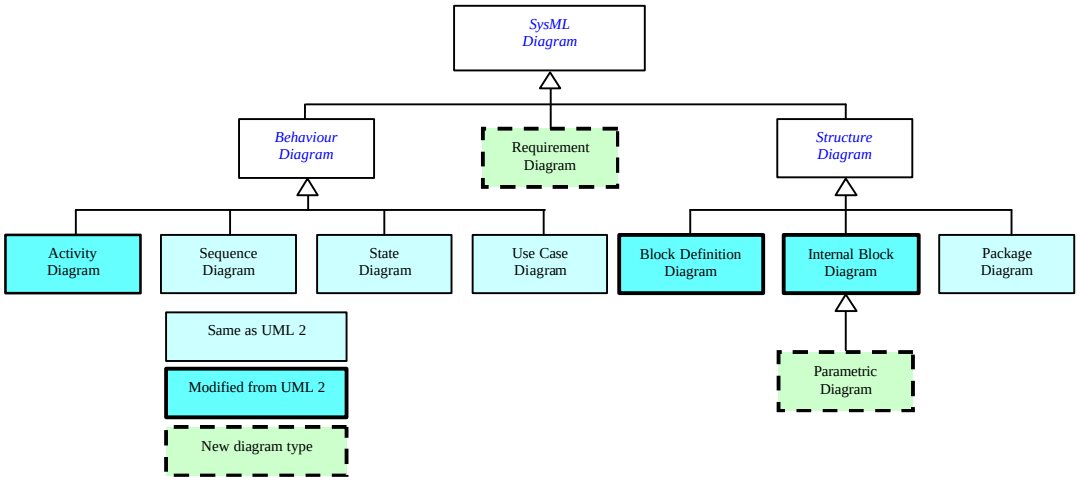

Figure 3: OMG SysML Diagram Taxonomy

OMG SysML includes diagrams that can be used to specify system requirements, behaviour, 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 behaviour 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. OMG 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. An example of these different diagrams for an ABS system is shown in Figure 4.

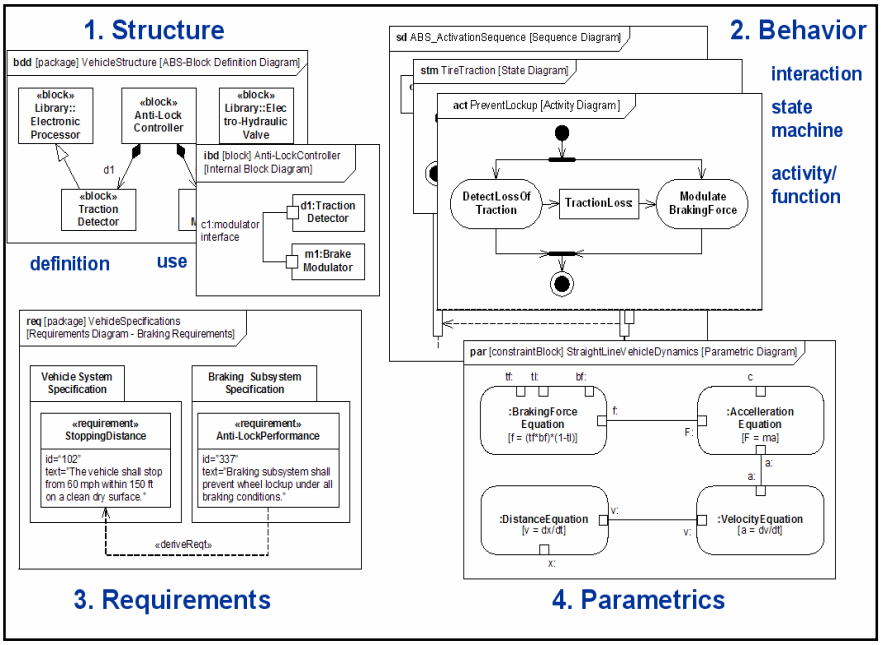

Figure 4: The Four Pillars of OMG SysML

This list of diagrams is not exclusive. If a modeller feels, for example, that a UML communication diagram is necessary to communicate a particular concept, or requirement, this will be allowed.

Each OMG SysML diagram has a frame with a contents area, a heading and a diagram description as shown in Figure 4. The frame is a rectangle that is required for OMG SysML diagrams (Note: the frame is optional in UML). The frame can designate a model element that is the default namespace for the model elements enclosed in the frame. A qualified name for the model element within the frame must be provided if it is not contained within the default namespace associated with the frame. The top level "Model" name is the highest level namespace for model elements. The frame may include border elements associated with the designated model element like ports for blocks, entry/exit points on state machines, gates on interactions, parameters for activities and constraint parameters for constraint blocks. The frame may sometimes be defined by the border of the diagram area provided by a tool.

Cross-Cutting Constructs. Figure 4 makes use of cross-cutting constructs. These apply to both structure and behaviour. Cross-cutting constructs support concerns that cut across the different views and may be addressed by all or disparate parts of the model. See Hause (2006) for more information. These constructs take the form of Allocations, Requirements and Parametrics. Allocations define a basic allocation relationship that can be used to allocate a set of model elements to another, such as allocating behaviour to structure or allocating logical to physical components.

Example Diagrams. The figures shown in this paper were created in Artisan Studio and were based on those shown in the SysML tutorial created by Sanford Friedenthal, Alan Moore, and Rick Steiner referenced in Friedenthal, et al, (2006).

REQUIREMENTS IN OMG SYSML

One of the two principal extensions to OMG SysML is support for requirements. The «requirement» stereotype extends class to specify the textual "shall" statement and capture the requirement id#. The requirement diagram is used to integrate the system models with text based requirements that are typically captured in requirements management tools. The UML containment relationship is used to decompose a requirement into its constituent requirements. A requirement is related to other key modelling artefacts via a set of stereotyped dependencies. The «deriveReqt» and «satisfy» dependencies describe the derivation of requirements from other requirements and the satisfaction of requirements by design, respectively. The «verify» dependency shows the link from a test case to the requirement or requirements it verifies. In addition, the UML «refine» dependency is used to indicate that an OMG SysML model element is a refinement of a textual requirement, and «a copy» relationship is used to show reuse of a requirement within a different requirement hierarchy. The «rationale» concept can be used to annotate any model element to identify supporting rationale including analysis and trade studies for a derived requirement, a design or some other decision.

Only the most basic attributes of a text based requirement are included in the OMG SysML specification. More specialized requirement types can be designated using specialization of the «requirement» stereotype. Typical examples are operational, functional, interface, control, performance, physical and storage requirements. These stereotypes may restrict the types of model elements that can satisfy or refine the requirement. For example, perhaps a «performanceRequirement» can only be satisfied by a set of Constraints in a parametric diagram along with an associated tolerance and/or probability distribution, or a «functionalRequirement» might be satisfied by an activity or operation of a block. A (non-normative) set of such specialized stereotypes can be found in Appendix C of the OMG SysML specification.

The requirements model can be shown in graphical, tree structure or tabular format. The graphical format is called a Requirements Diagram--an example of which is shown in Figure 5.

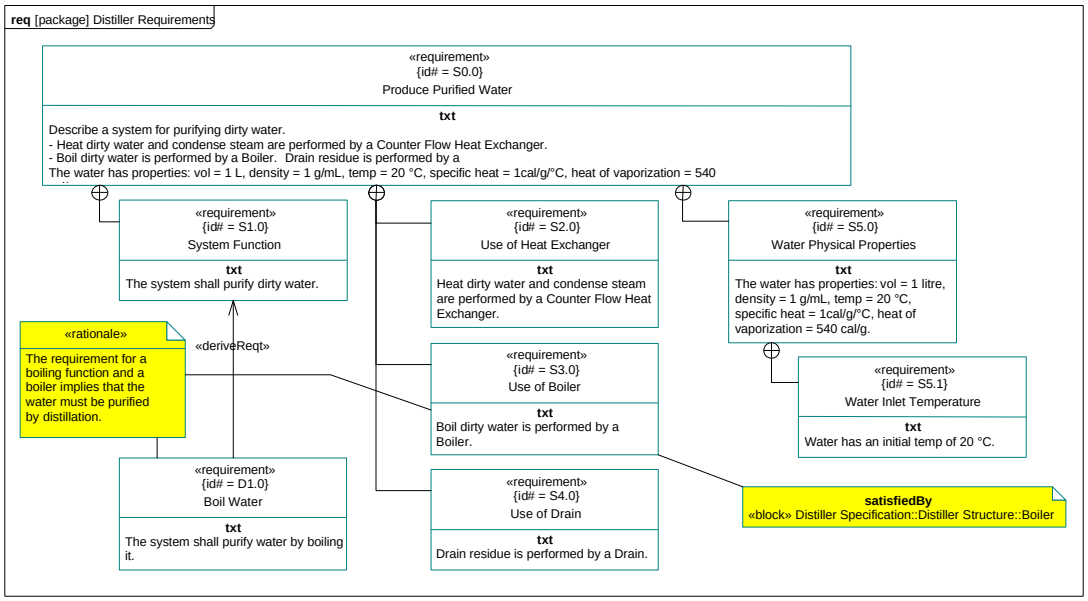

Figure 5: Requirements Diagram

Figure 5 shows the requirements flow down for a hypothetical system. For example, requirement D1.0, Boil Water, has been specified to be derived from system level requirement S1.0 by using the «deriveReqt» dependency. Requirement S0.0 is shown to be decomposed into a set of sub-requirements, S1-S5 that may illustrate other relationships such as «trace», «refine», «verify» and «satisfy». For example, an attached note indicates that requirement S3.0 is satisfied by the «block» Boiler. Each requirement has a unique identifier shown next to the id# tag and the text of the requirement is shown in the compartment labelled "txt" in the lower part of the class box. The format shown is only one of a number of possibilities and ergonomic profiling can be used to modify the display format to correspond to the needs of the audience. The exact capabilities for this type of feature will depend on the tool being used. (ARTiSAN Studio 6.1 is used in these examples).

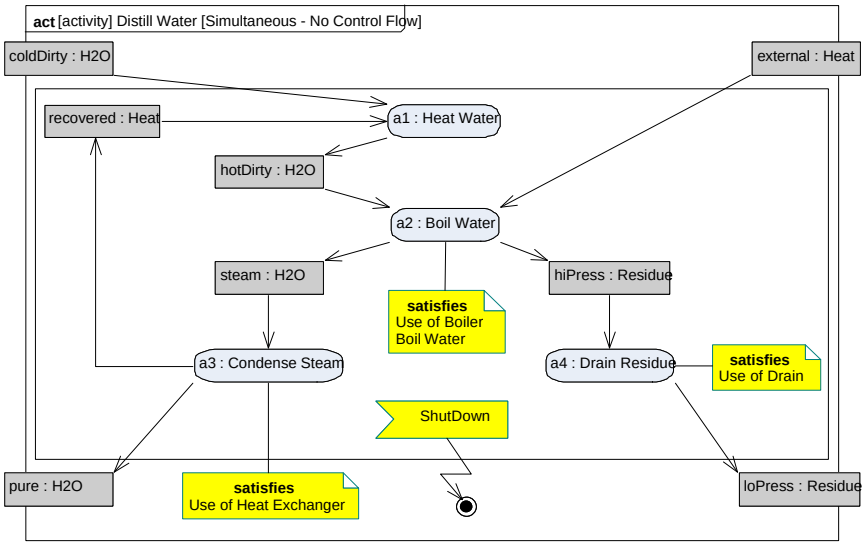

Figure 6: Distil Water Activity with Requirements

Additionally, requirements and their relationships can be shown on other diagrams such as this activity diagram depicted in Figure 6. The activities have been linked to the requirements they «satisfy» and the relationship is shown in an attached note. For example, activity a3: Condense Steam satisfies the requirement Use of Heat Exchanger.

The requirements model is not meant to replace external requirements management tools, but is meant to be used in conjunction with them to increase traceability within UML models. It could also be used for modelling the requirements and system for smaller projects. Its great advantage is that it allows for the modelling of the requirements, the system, and the traceability between them to be performed in a single model. Additionally, tools that provide the functionality to import, export, and synchronize requirements and their relationships between the OMG SysML model and an external requirements management tool will allow developers to perform requirements traceability in the tool while taking advantage of the features provided by specialist requirements management tools (Hause & Thom, 2005).

STRUCTURE

The major structural extension in OMG 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. An OMG 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. OMG SysML provides standard ports which support client-server communication (e.g., required and provided interfaces) and FlowPorts 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 is shown in Figure 7.

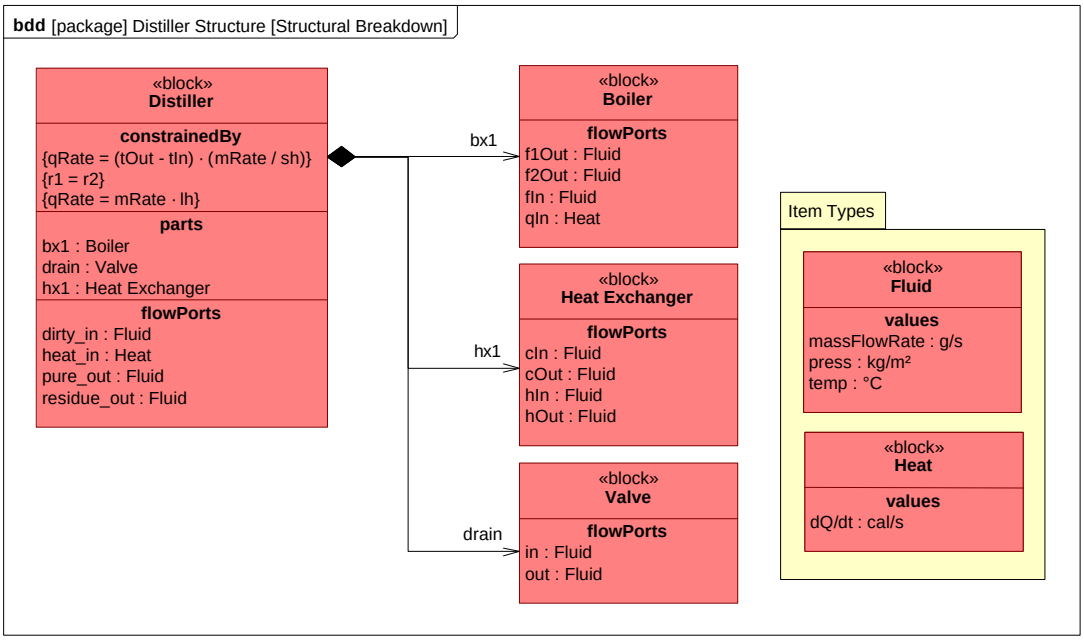

Figure 7: Block Definition Diagram

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 FlowPorts compartment describes the interface to the block.

A simple example of an internal block diagram (ibd) is shown in Figure 8.

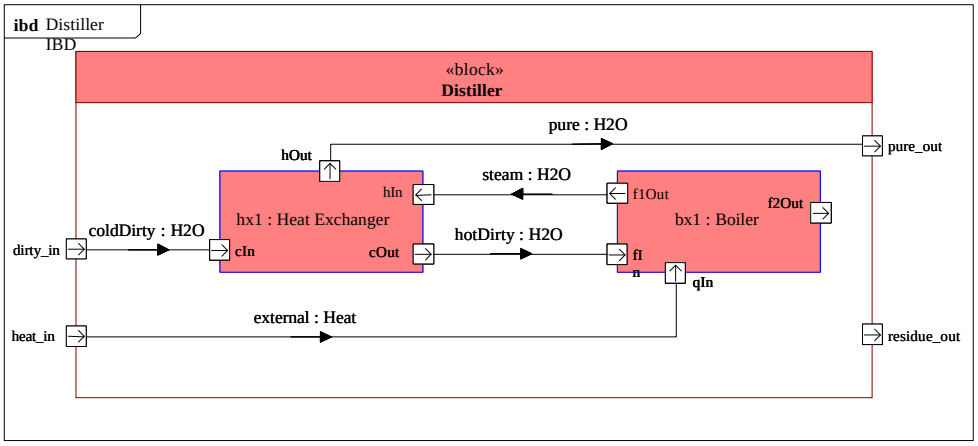

Figure 8: IBD for Distiller

Figure 8 shows the internal structure of the Distiller block. It shows two parts (the drain Valve is not shown)—hx1, a Heat Exchanger and bx1, a Boiler—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 as shown in Figure 9 in the Parametric Models section of this paper.

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.

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. 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. <>). The fill colour of the square is white and the line and text colours are black, unless the FlowPort is conjugated in which case the fill colour of the square is black and the text is in white. A conjugated FlowPort is the port at the opposite end. In other words, the flow specification may define a set of items flowing in and out. At the other end, the directions for these will be reversed.

PARAMETRIC DIAGRAMS

Parametric diagrams are used to describe constraints on system properties to support engineering analysis. In order to support this type of modelling a ConstraintBlock has been introduced into OMG SysML. A ConstraintBlock defines a set of parameters and one or more constraints on the parameters. By default, these parameters are non-directional and so have no notion of causality. These ConstraintBlocks are used in a parametric diagram to constrain system properties. ConstraintBlocks may be used to express mathematical equations such as ‘F=m·a’ and ‘a = dv/dt’, or statistical values and utility functions such as might be used in trade studies. Based on the reusable concept of a block new ConstraintBlocks can be built by reusing more primitive ConstraintBlocks such as basic mathematical operators.

OMG SysML also defines a model of value types that can have units and dimensions and probability distributions. The value types are used to type properties of blocks.

The Parametric Diagram is a specialized variant of an internal block diagram that restricts diagram elements to represent constraint blocks, their parameters and the block properties that they bind to. Both parameters and properties may be represented as small “pin-like” boxes to help make the diagrams more scaleable. An example of the Parametric Diagram is shown in Figure 9.

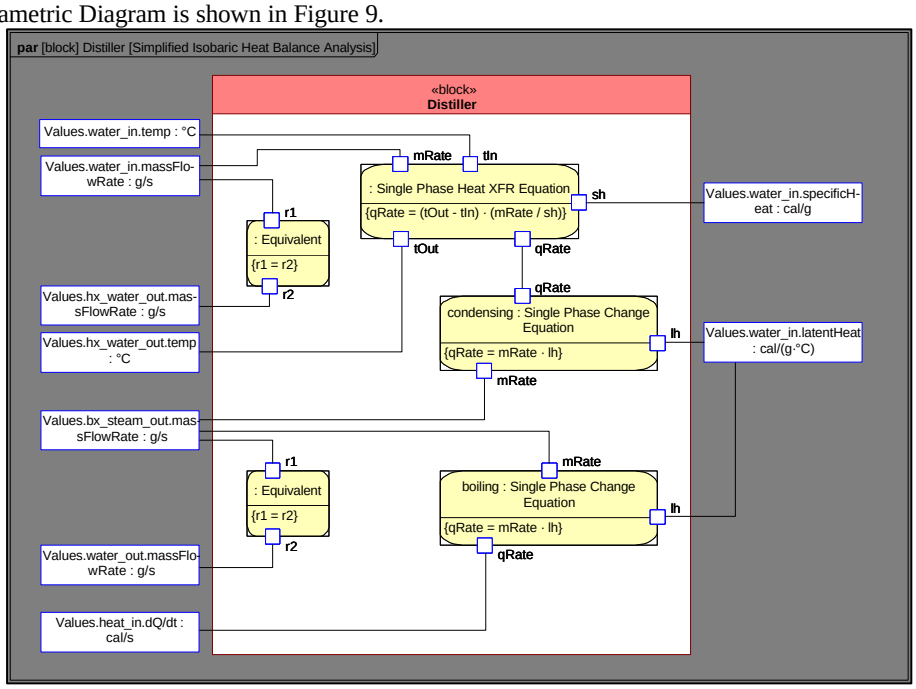

Figure 9: Parametric Diagram

The parametric diagram in Figure 9 describes the constraints on fluid flow through the Distiller block. The nodes on the left and right sides represent the item flows on Figure 8 and the five constraints (yellow rounded rectangles) show how the parameters of the various generic and specific equations bind to properties of the item flows to enforce the flow constraints. Equations may be shown using the callout notation as attached notes. Alternatively, a compartment in the constraint symbol may be used, as shown in this example. Note that the flow properties have types with defined units (part of the Profile’s Non-Normative Extensions).

BEHAVIOR

The OMG SysML behavioural diagrams include the activity diagram, sequence diagram, state machine diagram and use case diagram. State machines are used to specify state-based behaviour in terms of system states and

their transitions. Sequence diagrams describe message based behaviour of the system, subsystem, activity, use case or other owning construct. Use cases describe behaviour in terms of the high level functionality and uses of a system that are further specified in the other behavioural diagrams referred to above. These diagrams remain virtually unchanged from UML 2. For further information, the reader is referred to OMG (2003a), OMG (2003b), and Hause (2002). Activities, which have been significantly extended from UML 2.0 activities, represent the basic unit of behaviour that is used in activity, sequence and state machine diagrams. The activity diagram is used to describe the flow of control and flow of inputs and outputs among actions.

CONTINUOUS SYSTEMS

Additionally, OMG SysML provides extensions that might be very loosely grouped under the term "continuous" but, are generally applicable to any sort of distributed flow of information and physical items through a system. These are:

- Restrictions on the rate at which entities flow along edges in an activity, or in and out of parameters of a behaviour. This includes both discrete and continuous flows, either of material, energy or information. Discrete and continuous flows are unified under rate of flow, as is traditionally done in mathematical models of continuous change where the discrete increment of time approaches zero.
- Extension of object nodes, including pins, with the option for newly arriving values to replace values that are already in the object nodes. OMG SysML also extends object nodes with the option to discard values if they do not immediately flow downstream. These two extensions are useful for ensuring that the most recent information is available to actions by indicating when old values should not be kept in object nodes, and for preventing fast or continuously flowing values from collecting in an object node, as well as modelling transient values, such as electrical signals.

Probabilities. OMG SysML introduces probability into activities as follows:

- Extension of edges with probabilities for the likelihood that a value leaving the decision node or object node will traverse an edge.
- Extension of output parameter sets with probabilities for the likelihood that values will be output on a parameter set.

Allocations. OMG SysML includes an allocation relationship to allocate one model element to another. Allocation is the term used by systems engineers to denote the organized cross-association (mapping) of elements within the various structures or hierarchies of a user model. Often this is the allocation of function to form, such as the deployment of software on a hardware platform, or a use case to an organization or system entity. From a systems engineering perspective, this is applicable to abstract system specifications rather than a particular constrained method of system or software design. Allocations can be used early in the design as a precursor to more detailed rigorous specifications and implementations. The allocation relationship can provide an effective means for navigating the model by establishing cross relationships and ensuring the various parts of the model are properly integrated. The OMG SysML specification includes some specific subclasses of allocation for allocating behaviour, structure and flows, but these are given more as examples rather than an exhaustive list. A typical example is the allocation of activities to blocks (e.g., functions to components).

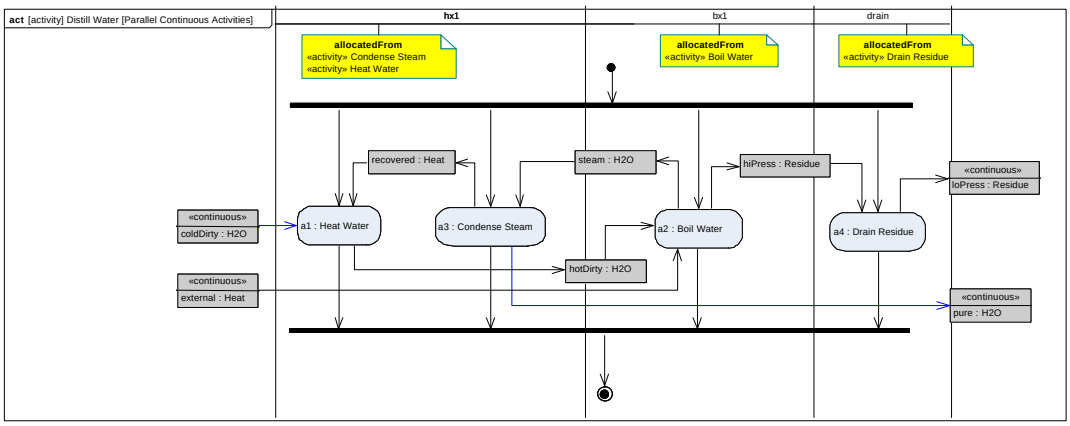

Figure 10: Activity Diagram Allocation

Figure 10 shows an example activity diagram with activities located in swim lanes. The placement of the activity within the swim lane is a type of allocation. It indicates that the system element associated with the swim lane is responsible for performing the activity. This shows allocation at the highest level of abstraction.

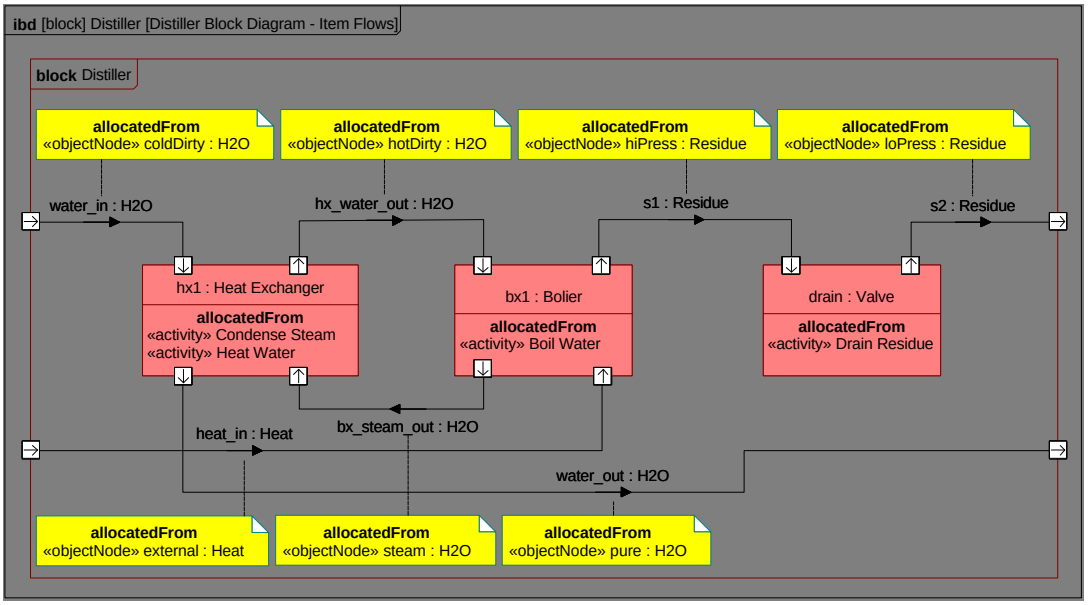

Figure 11: Structure Diagram Allocation

Figure 11 shows allocation on the Internal Block Diagram. The allocation of activity invocation to parts is shown in the allocatedFrom compartment. The allocation of activity object nodes to Item flows is shown via call-outs (the boxes shown in Figure 11 labelled "allocatedFrom" and linked to the flows. Again, it is worth pointing out that these are some examples of allocation and that others will be necessary depending on the scope of the systems engineering activity and the domain.

Finally, allocation can be shown in tabular form, as well. This, of course, is only possible if the model has been built on a database and is not simply a series of unrelated diagrams.

MODEL INTERCHANGE

AP233. The OMG SysML specification is intended to be compatible with the evolving ISO AP233 standard. AP233 is a data exchange protocol for systems engineering data based on ISO 10303. STEP (ISO 10303) is a standard to describe, represent and exchange industrial data in a computer interpretable format. For data exchange between different application domains, a neutral (i.e. tool independent) data model has to be defined. This is called AP (Application protocol). ISO 10303 provides application protocols for representing this data exchange. Currently around 40 different APs are defined. The intent of AP233 is to support the whole system development life cycle ranging from requirements definition to system verification and validation. Different application areas include: Engineering Analysis, Algorithm Design, Planning Tools, Testing Tools, Software Design, Mechanical Computer Aided Design (CAD), and Electrical Computer Aided Engineering (CAE). This goal is consistent with that of Systems Engineering in general, which is to tie together the different domain engineering disciplines into one consistent view of the system. Figure 12 shows how OMG SysML is intended to share data with other engineering tools using AP-233 as the neutral data exchange format.

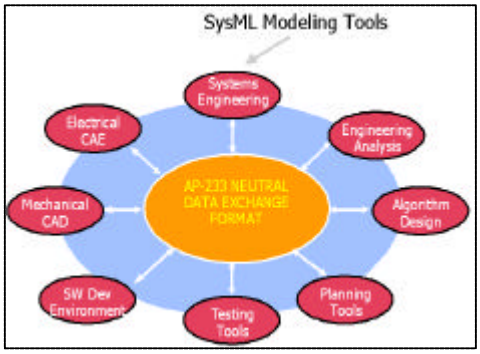

Figure 12: AP-233

The AP233 OMG SysML alignment activity is performed as part of the definition of OMG SysML to ensure that:

- OMG SysML models can be exchanged using the AP233 data exchange protocol.
- Data available in AP233 - being generated by another tool - can be visualized in an OMG SysML tool.

XMI. Another option to exchange OMG SysML models is XMI (XML Metadata Interchange). Because OMG SysML is defined as an extension of the UML metamodel, OMG SysML models can also be exchanged using an XMI schema. There are however, some limitations to XMI. For example, the interchange of diagrammatic information will be covered by a different specification.

Data Interchange Problems. Data interchange between modelling tools has been a long-term goal. The CDIF (CASE Data Interchange Format) family of standards developed in the 1990's defines a single format for exchanging information, potentially through the whole IS lifecycle, between CASE tools from different vendors. Some work was done in an effort to exchange data between Yourdon Data Flow Diagrams and UML models. This quickly ran into problems because the underlying theory of the two systems, Data Flow Analysis and Object Oriented Design, are so different. For a particular model a specific Function block and class may represent the same concept; this will not generally be the case. Also, the concept of encapsulation involves grouping data and functionality together in a single entity, something that cannot be modelled in Yourdon. As a consequence, translating from one to another will involve a loss of information. This was similar to early translation programs that translated the quote, "The spirit is willing, but the flesh is weak" from English to Russian and back again. This yielded the phrase, "The liquor is good, but the meat is awful." Any information format that will be used to translate from one set of concepts to another must by necessity contain a superset of the concepts potentially found in all systems. That will make the task of translating models very difficult indeed. Prototypes have already been built by ARTiSAN Software Tools and Eurostep to exchange requirements and architecture information. However, this challenge is intended to be addressed through the use of MOF-based models that are being developed through the OMG.

CONCLUSION

Three years from the initial Request for Proposal, version 1.0 of OMG SysML was approved for finalization in April 2006. The extensions made to UML 2 in the OMG SysML Profile will provide Systems Engineers with a robust modelling language for modelling systems that include models of requirements, behaviour, structure and parametrics. At the same time, the profile's extensive reuse of UML will facilitate a much smoother flow down from systems engineering to software engineering than otherwise possible. For further information, the reader is directed to the OMG SysML v1.0 draft specification available at the OMG system engineering site at http://omgsysml.org/index.htm.

REFERENCES

Dijkstra, Edsger, 1974, On the Role Of Scientific Thought, accessed online November, 2005 from Hause, M., 2006, Cross-Cutting Concerns and Ergonomic Profiling in UML/OMG SysML, July 2006, INCOSE International Symposium 2006 Proceedings.

Friedenthal, S., Burkhart, R. "Extending UML From Software To Systems," Proceedings Of The INCOSE 2003 International Symposium, 2003.

Friedenthal, S., Moore, A., Steiner, R. "OMG SysML™ Systems Modelling Language Tutorial", [online] Available from: http://www.omgSysML.org [Accessed July 2006].

Hause, M., 2001, Rebuilding the Tower of Babel – The Case for UML Extensions, May 2001, INCOSE UK Chapter – Spring Symposium 2001 Proceedings.

Hause, M., 2002, Using Use cases and Scenarios as Part of a Safety Case, May 2002, INCOSE UK Chapter – Spring Symposium 2002 Proceedings.

Hause M., Thom F., 2003, Building a Systems Engineering Ontology Using UML, INCOSE International Symposium 2003 Proceedings.

Hause, M., Thom, F., Modelling High Level Requirements in UML/OMG SysML, July 2005, INCOSE International Symposium, Rochester 2005 Proceedings.

Lykins, H., Friedenthal, S., And Meilich, A.. Adapting UML For An Object-Oriented Systems Engineering Method (OOSEM)", To Appear In Proc Tenth Annual Int Symp INCOSE. Minneapolis, Mn, USA, July 16-20, 2000.

Object Management Group (OMG), 1997. Unified Modelling Language: Semantics 1.1 Final Adopted Specification ptc/97-08-04. [online] Available from: http://www.omg.org [Accessed September 1998].

Object Management Group (OMG), 2003a. Unified Modeling Language: Superstructure version 2.0 Final Adopted Specification ptc/03-08-02. [online] Available from: http://www.omg.org [Accessed September 2003].

Object Management Group (OMG), 2003b. Unified Modeling Language: Infrastructure version 2.0 Final Adopted Specification ptc/03-09-015. [online] Available from: http://www.omg.org [Accessed September 2003].

OMG SysML™ Object Management Group (OMG), 2003, UML™ for Systems Engineering Request for Proposal OMG Document: ad/03-03-41, Available from www.omg.org. [Accessed September 2003].

Object Management Group (OMG), 2006, *Systems Modeling Language version 1.0*, Available from www.omgsysml.org. [Accessed July 2006].

BIOGRAPHY

Matthew Hause, Chief Consultant - Artisan Software Tools. Matthew has been developing real-time systems for almost 30 years. He started out working in the Power Systems Industry, and has been involved in Process Control, Communications, SCADA, Distributed Control, and many other areas of real-time systems. His roles have varied from project manager to developer. His role at Artisan includes mentoring, sales presentations and training courses. He has written a series of white papers on project management, Systems Engineering, and systems development with UML. He has been a regular presenter at INCOSE, the IEE and other conferences.