Compilation of SysML RFI- Final Report

Systems Modeling Language (SysML) Request For Information OMG Document: syseng/2009-06-01

Dr. Robert Cloutier
Mary Bone
Report date: 02/20/2010

STEVENS Institute of Technology

SSE School of Systems&Enterprises

Table of Contents

Preface

This document includes the entire set of textual responses to the questions contained in the responses to the SysML Request for Information (OMG document number syseng/09-06-01). Added to most questions is an analytical view of data that lends itself to graphs. Preliminary analysis (syseng/09-12-04) of this data was presented to the OMG SE DSIG on December 8, 2009 in Long Beach, and again at the INCOSE International Workshop in Phoenix, AZ in February.

Minor edits were performed on the dataset to improve readability. These edits include correction of some obvious spelling errors, and deletion of comments like N/A, or not used, or no comment. If a participant responded with "No Comment, use all the time" the comment was not deleted. While some effort was made to correct capitalization errors, we may have missed some. Since the intent of this report is to provide the raw data from the survey, no effort was expended in attempting to correct grammatical errors of the responses.

Question 1: Demographic Information

Removed to protect company and individual interests

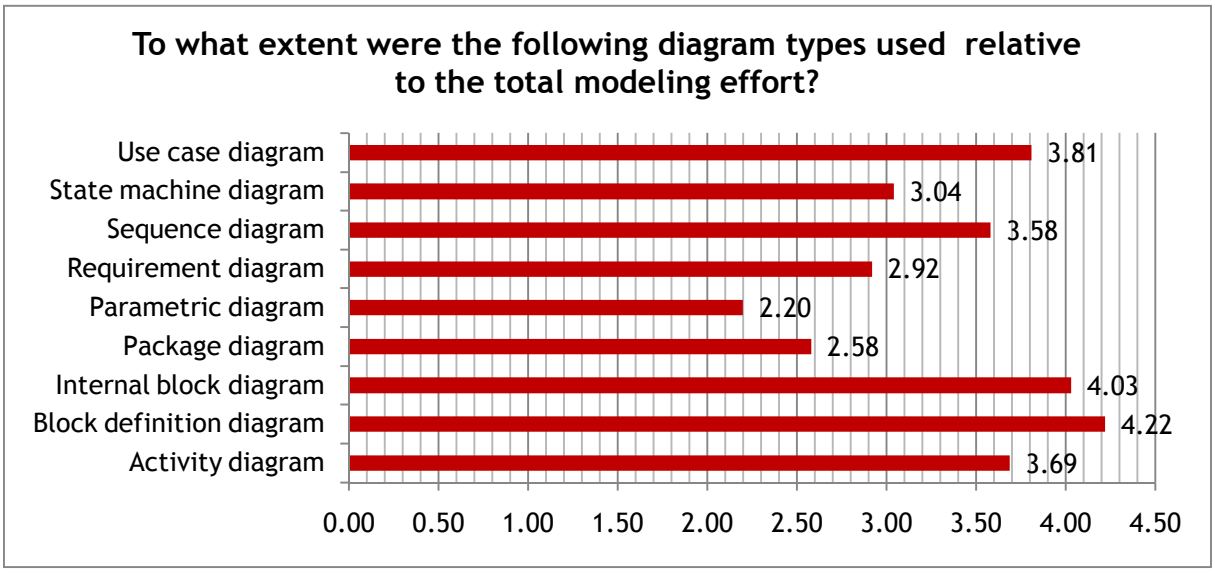
Question 2: Demographic Information

Removed to protect company and individual interests

Question 3: To what extent were the following diagram types used relative to the total modeling effort?

Results

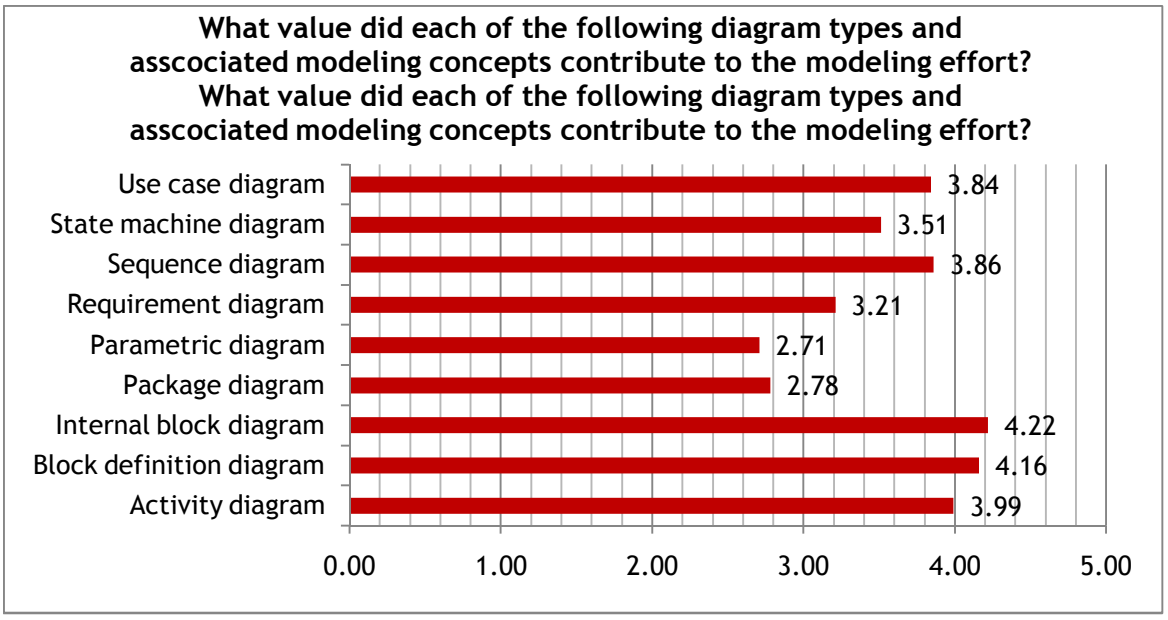
Where 1= Low Use 3= Medium Use and 5 = High Use



Question 4: What value did each of the following diagram types and associated modeling concepts contribute to the modeling effort?

Results

Where 1= Low Value 3= Medium Value and 5 = High Value



Open-Ended Responses

Identify other features you found useful that are not listed above (e.g., allocations):

- Requirement Relationships, Relationship Matrix, Notes to expand use cases in activity diagrams, boundary, control, entity analysis blocks
- Information Flow, Ports, Composite Structure
- Standard ports, required/provided operations, connectors.
- Allocations are most useful indeed
- Interfaces
- Allocations
- Artisan Narrative for their Sequence Diagram. Animation of Sequence Diagram worked well with Fire Control SMEs
- Timing*, frequency, phase, and stability (bifurcation) diagrams not in SysML

- export of the model to XML is extremely important to our effort
- Use Case to Block and Part allocations
- Corba toolbar
- allocations, traceability links
- Allocations most certainly. Stereotypes; relationships;
- stereotyping, inheritance
- tabular format
- Allocations, Viewpoints, Views, Value Types, Physical Dimensions and Units
- various matrices and tabular requirement diagrams provide high value also
- Content diagram (MagicDraw) to create navigation aids within model
- Table and Matrix views notations, Allocations are being used but perceived as somewhat informal.
- High value of the requirements model (but not the diagram)
- Generated trace and allocation table views
- structure in "browser" section
- Activity swimlanes, allocations, relating requirements to things besides other requirements, UML timing diagrams are useful as well
- Allocations: Medium value
- Allocations: behavior->block, block->block (deployment);
- Allocation of requirements to model elements
- allocations esp. for tracing requirements to structure and similar relationships
- Context, Architectural, Concurrency and Constraints diagrams
- Metamodels (e.g., UML4SysML) and their corresponding profiles (e.g., SysML)
- too many to list
- Too much emphasis is put on the diagrams. Some of the most significant value is the network of relationships that the model as a whole represents. Slices of these relationships can be shown via one or more of the diagram types. We have been working to develop query utilities that allow relational threads between model elements to be traced and reported on in order to support change impact assessments and interface compatibility assessments.
- Allocations!
- The relationships such as satisfy and allocate were very useful
- Stereotype, constraint
- Flow Ports not well defined;
- stereotypes, derivations, containment
- Ports, Constraint Blocks
- Cross Diagram Methodology (Linking sd with act diagrams, uc with req, bdds with ibds)
- Cross-Layer (for Ad Hoc Networks)
- Expect high value when we implement all of the checked diagrams. We are concentrating on the sequence diagrams.
- Ports
- Have attended SysML course, but have not yet used SysML. I therefore have no modeling experience to report as yet.
- satisfy, allocations, generalization, composite

- Allocations, Flow Ports
- Have used these diagram types as UML 2.x have not used SysML

Question 5: What issues did you have with each of the following diagram types and associated modeling concepts? Please include in your explanation the modeling constructs that contributed to the issues (e.g., use of ports on an ibd). Note: Additional space is provided at the end of the survey in question 60 if needed.

Open Ended Responses

Activity diagram

- No issues. Generally intuitive.
- No notion of time. It is hard to group activities into a swim lane to allocate. Allocate swim lanes are not the same as the item that it represents. Why aren't they just a block that looks like a swim lane? I add an activity it is allocated directly to the block and not to another item that represents the block. Where is the action semantic language to give behaviors to the activities?
- Passing of objects within the diagram and associating those with classes
- Problems with rules against using forks, joins, and connectors
- Action activation semantics; control vs. object flows
- The actions are categorically descriptive, but the actual mathematical description of behavior is often underdetermined.
- Activity diagrams were used for advanced algorithm descriptions as a design aid to SW engineering. : 1 There is disparity between different tools in the implementation of ADs.: 1 Tool issue - Rhapsody pre-7.5 doesn't adequately implement activity diagrams.: 1 What format to use. The context or perspective of the activity I was trying to describe.: 1
- Activity diagrams should only have actions, not events, message objects, object flows, etc.
- Conversion to BPMN
- When activities are placed in swimlanes that are typed with a block the block does not own those activities.
- Swim lanes are not real "blocks" - just picture things. MUST be same as blocks on other diagrams (drag a block from a BDD onto an activity diagram should result in a swimlane);
- Easy for people to gloss over the token semantics and hence abuse the model as just a pretty picture
- Not clear how to use in system engineering
- Pin/argument cross-identification for InvokeActions
- No standardized formal semantics for guards and conditions
- Activity parameters can't be connected to Block ports, parameters can't be used in parametrics. Action-role mapping in activity decomposition diagram is weird, synchronization is not described.

- Activity Diagrams are mainly used to describe system level processes as interactions between subsystems and stakeholders. Our tools only partially support the simulation of activity diagrams and we are not aware of any SysML tool that provides complete support for simulating SysML activity diagrams. It seems that current end users that need to simulate continuous behavior and transformational synchronous behavior turn to dedicated tools using proprietary formalisms because of lack of tool support for ACD simulation by the SysML tools. Also, the additions to support continuous systems and probability made by SysML to UML, might make sense descriptive but still add to complexity of the language that is already perceived as being too complex.
- Naming of multiple usages (parts, ports with multiplicity higher than 1) is not consistent with IBD - standard and tools are vague - causes many interpretation/usability problems to end-users
- Can't use constraints and property parts in conjunction with activities, cant be able to put instance values on actions, call behavior actions and buffers etc, cant show contents of activity diagrams the same way IBDs do (i.e. - internal structure), cant share actions call behavior actions etc across other activities to make cross cutting functional descriptions through references or "dot" notation like IBDs (maybe more of a tools issue but spec isn't very clear), can't show states and associate them with activities.
- A view of deeply nested activities is terribly missing. The same structured, "look inside" view, as in IBDs, would help enormously in avoiding having too many diagrams.
- Great difficulties in keeping parameters, parameter nodes, and pins in sync. In particular for the <<rate>> attribute.
- Consistent stereotyping of Pin, Parameter node, and Parameter is not defined.
- More than the <<rate>> attribute is needed for control systems, e.g. duration, complexity, latency, jitter, clocked, etc.
- Inheritance of parameters is not clearly specified. e.g. if new parameters should go at the beginning or the end of existing ones. Furthermore, the situation is complicated by parameters, pins, and parameter nodes which creates a hell lot of work for tool-vendors.
- Activities and Parametrics are two sides of the same coin. The former describes behavior in causal manner, the latter in a-causal manner. When describing the behavior of a system they appear closely related and you would like to constrain attributes/parameters of actions
- Activities together with static system properties.
- Support for token generation on outgoing vertices. Need to model multiple instances of an execution thread
- Other than difficulties with making details work in the tools, nothing special
- Use of Signals
- deferred to question 60 and/or rtf responses done or to be done
- It's difficult to specialize activity diagrams; e.g., control flow only; data-flow only; workflow. It's non-trivial to use activity diagrams to show orchestration & choreography properly. Sometimes, it would be nice to show explicitly the type information on an object flow edge without adding a central buffer node.
- No explicit way to link activities and actions to operations, stores to properties, activity parameters to parametrics.
- Easy to get cluttered and confusing.
- Understanding the nuances of pins and control objects;
- Designers tended to make these overly complex and to perform inappropriate subsystem

design.

- Linkage between AD, BD, SD, UCD and requirements was minimal which allowed them to become inconsistent
- The basic construct of the activity diagram is intuitively clear to engineers. The more sophisticated use of data pins, and their relationship to ports and data types elsewhere in the model requires lengthy explanation, but is a tremendously powerful utility to tie the operational or functional behavior as represented in an activity diagram to the physical and logical interfaces defined in IBDs.
- Inability to decompose actions from one activity diagram to another.
- Poor support by tools
- General: Need more detailed SysML definitions to avoid wide variations in usage such as messages and signals.
- Need to include the remainder of UML activity diagrams to allow SE to start system software architecture; make it clear/possible on how to access activity properties/behaviors; make sure activity properties are accessible in parametrics; need to support non-interruptible regions
- No defined semantics, little value in comparison to effort to produce
- Steep learning curve for the tool and difficulty dealing with complex projects (visual clutter.)
- Merge the concepts of Petri nets more closely with this. These should be executable. Syntax/semantics should be in the standard but they are not!
- Activity diagram was not used on last program
- Improve methods for associating requirements with activity blocks
- Similar to UML
- Crossing control and object flows; pins and port consistency ;
- Not sure why there is both a sequence diagram and an activity diagram. can't these be combined? Also, need a timing diagram mode (to micro seconds).
- Understanding multiplicity combined with I/O with actions and call behavior vs. actions
- I had difficulty using a tool (Rhapsody) that was not compliant with the spec. After awhile you confuse what the tool allows versus what the spec contains. Now that they have just made major updates, I need to re-learn.
- Significant problems with the difference between the activities and actions. Most of what I do is at the SoS level so actions do not apply which complicates the decomposition of activities.

Block definition diagram

- The close relationship with the UML Class diagram - almost redundant - just elide IBD
- The term "Block" is extremely cumbersome, particularly when the object classes, like information, artifacts, etc. are involved. For this, I prefer to use the UML constructs (which my SysML tool allows). I would have preferred if the <<block>> stereotype had been left out, and the <<class>> been maintained. This would not have affected the existence of stereotypes for SoS, System, Subsystem, etc.
- Hard to create data dictionary information with this diagram. Data stored in Tags is not easy to display. I would like SysML to include a form to fill out to populate blocks with the right info and to visualize what the info is in a block.

- Every operation used on an interface must be defined at every interface boundary (doubles the amount of work). Needs to inherit and allow modification.
- Useful to describe blocks and connections between parts of different packages.
- How to represent block instances?
- The block diagram can be extended to provide contextual interface to the block
- None: 2 BD diagrams mostly needed in OO system design. IBD's were used mostly.: 1 Training - SE's typically don't understand bdd's right away.: 1
- Security
- I'd like to be able to show ports on BDDs
- Interfaces definition.
- Not clear how to use ports.
- Reuse of system part design
- Desire for this to be strongly related to a functional block diagram was not realized
- No problems - except teaching people the difference between this and IBDs. Too often, folks want to build an IBD on the BDD!
- Not clear how to use aggregation (full diamond vs. empty diamond)
- Property specific type approach is unusable.
- BDDs and IBDs are the most used diagrams in SysML however we think that we need to distinguish between "advanced" elements such as GeneralizationSet or Multibranch associations and "basic" elements like Block or ReferenceAssociation. Our experience with practitioners is that they would like to avoid tools on their palette that are hard to understand and they are not likely to use. Another issue we observed with practitioners is that sometimes the distinction between part and block is not clear.
- A textual notation for ports, i.e. a compartment like the parts, value, or references compartment / Unclear is the usage containment vs. composition
- Inheritance of properties and interfaces still behaves like a software class - a compiler is expected to copy the inherited portion for example, so it is unclear what inheritance means.
- Better description needed on how constraints on blocks relate to constraints in parametrics. / Better description needed for the difference between a normal part property and a part property with an association is. / Value Types are weird as they need Unit and Dimension but Unit also has a Dimension attribute. And, as value types have a dimension and unit, they can have themselves attributes which are value types - how does this fit together? / Redefinition of Property-specific types are a hassle as the COMPLETE hierarchy has to be redefined (inherited) to change the definition of ONE single deeply nested part. Redefinition of default values of value attributes in a specialization hierarchy is cumbersome (redefined property). The same attribute has to be created in every specialization and then redefined. When defining a new default value in a specialization the value is also changed in the base class. / It is difficult for people to see what the difference between allocating an action/activity to a block and a block operation is.
- It is not easy to define domain specific models with special types of blocks, properties, etc. (see question 60)
- Confusion on use of associations in BDD vs connectors & ports in IBD
- Use of different types of associations (esp. shared and reference)

- Placement of ports is awkward due to limited flexibility in automatic label placement. e.g., most labels are shown "to the right" or "on top of" but in some cases, it would be useful to toggle the label placement to the left-of/right-of/above/below/... easily.
- Difficulty in knowing if additional information/compartments exist if hidden. Not necessary to see them all the time, but useful to know if they are defined.
- No issues; pretty well defined
- Linkage between AD, BD, SD, UCD and requirements was minimal which allowed them to become inconsistent
- The flexibility of BDDs allow for many different messages to be delivered. We have constrained our BDDs to only provide Directed Composition and Aggregation relationships that detail the system block decomposition. Having established this metastandard greatly simplifies BDDs and makes them intuitively, though open diamond vs. closed diamond notation often trips people up.
- Dropping of qualifiers and similar minor UML features make teaching SysML more difficult to UML users and doesn't really simplify the language
- No defined semantics, little value in comparison to effort to produce
- Steep learning curve for the tool and difficulty dealing with complex projects (visual clutter.)
- Block definition diagram were never baselined
- The difference between aggregation and directed composition is confusing. Also, the term "part" is confusing; it would be better labeled as "an instance" of a block.
- Functional decomposition using system blocks
- Linkage to class diagrams
- Multiplicity and not having instances (roles vs. instances)
- Used extensively, no issues.
- Most of my issues are not with the spec but trying to use a particular implementation. bdds are pretty intuitive.
- Had issues showing what data was communicated between blocks at different levels

Internal block diagram

- The very close relationship with the UML Class diagram
- It does not seem like consistency between diagrams is enforced.
- Doesn't seem to be a clean way to instantiate combinatorial logic (unclocked) operations
- Very useful for white box analysis.
- Flow ports. would like to make instantiations of fp or type with a specialization of a flow spec to show context specific use: 1 Good and clear examples were needed.: 1 No issues, used for block diagrams and associated interfaces and data flows: 1 none - very intuitive diagram: 1 Port names (in Rhapsody) cluttered the diagrams and frequently drew questions from non-SysML reviewers.: 1
- Hight SAFE
- Flow ports should only be used for analog messages, not for computer messages (what I see most trying to model)
- My tool (EA) prevented me from drawing ports on the diagram frame
- Port definition. Association with other diagrams.
- Not clear how to use ports. Connectors that are port specific.

- Not clear representation and semantics
- Desire for this to be strongly related to a functional block diagram was not realized; interface representation seemed overly cumbersome
- Lack of useful "abstraction" concepts - for example: primary function vs. services vs. foundation. To be semantically correct, you have to include all levels on the diagram, but we want to hide certain items on a high-level diagram (make an object to appear as a node (dot on the line) or as a line itself (lines can represent objects, too).
- We need better support for property specific types (redefinition in general) -- crucial for reusable libraries
- Difficulty modeling high level data flows before deciding\defining ports and precise interfaces.
- Nested ports are required. Can parts with multiplicity more than 1 be represented as more than one box (to be connected with different parts)?
- Lack of inherent language support for nested ports, can't elide outermost ports in a delegation "stack" (Q60)
- It should be possible to hide the internal parts of a port in an ibd without losing the visual information that they have connections to the outside of the enclosing block / Ports could represent complex interaction points that have their own structure and behavior which is described by the typing block. That block itself could have ports which leads to nested ports. That is a very useful concept as proven by many projects.
- Can’t put instance values on parts, ports, value properties etc., no way to show contents of ports with a flow spec, can’t use actions, call behavior actions etc and states in on IBDs, flow ports and standard ports still have software specification and meaning – i.e. the still describe properties required of the implementing class – this depends on a compiler and is too specific for topics other than software.
- No proper syntax and semantics for delegation/junction ports. It must be possible to hide the internal parts of a part in an ibd without losing the visual information that they have connections to the outside of the enclosing part. / No proper syntax for discipline specific interaction points, i.e. ports for mechanics, optics, electronics, information / Difficult use of context-specific values due to dependency on UML InstanceSpecification. Some tool vendors help a lot, e.g. NoMagic with their implementation. However, there a different interpretations of the spec by different vendors. / Unclear or no description of nested ports in order to group ports, e.g. group together all electronic, all mechanical, all optical, and all information interaction points in ONE port for better re-use and maintenance. / Flow Properties CANNOT be connected individually but only the whole port.
- Difference between the flow and the connector item is difficult to understand
- Use of flow ports vs. standard ports
- The port notation for data direction is not clearly noticeable; especially the reverse indication
- Flow port semantics and binding to behavior needs to be clarified
- Easy to get confused by ports vs. flow ports.
- No input; didn't have a chance to use it yet
- The diagrams are very intuitive and familiar as they are similar to a standard Visio product. Significant discussion and review is needed to elucidate the broader functionality and relationships to interface specifications and requirements and how ports are tied to data pins in activity diagrams.

- Understanding that connections on an IBD are associated with a Block, and showing the same parts in a drawing associated with another block those connections are not made by populating.
- Nested ports needed; ports that are reusable parts; Unclear how/if multiple different configurations can be done without repeating the bdd; mapping of connectors to associations unclear, are connectors real or can they be conceptual; flow ports still not clear; property-specific values not sufficient(or workable); do parts used in different ways require separate relationships on the BDD
- Ports on IBD
- No defined semantics, little value in comparison to effort to produce
- Steep learning curve for the tool and difficulty dealing with complex projects (visual clutter.)
- Probably a tool issue. I can't hide the type on a flow spec. This wastes screen real estate.
- IBDs become very difficult to manage/traverse once lower level IBDs are developed. Some type of "hyper-text" or linking on a single block to the next level of derivation would be very worthwhile.
- Ability to inherit ports from higher level system blocks, so they do not need to be recreated
- extension level of bdd
- port specification consistency checking, crossing connectors
- Multiplicity and instances of parts - seems illogical not to have instances
- Interface blocks do not support two-way messaging
- Used extensively, no issues.
- I guess the subtle difference between a UML object and a SysML part. It's easier to explain an object instance versus a parts role. Tool implementation can confuse this as well. How multiplicity is handled as well.
- Interfaces and ports - how to represent message based interfaces

Package diagram

- Organizational
- Largely redundant with bdd's: 1 none: 1 Not Used on our program: 1 organizing with Package diagrams was difficult: 1
- Never user. The tool project browser creates package hierarchy and object containment, so this diagram is never used - no value added.
- package is a foreign concept to our system engineering processes
- View and viewpoint implementation is problematic, as based on packages
- PKG are rarely used since the structure of the design can be expressed in a BDD and most tools have browsers that show them the division of the design. We think that SysML can do without this diagram.
- Sometimes just use BDDs, could be more useful if you could additionally develop diagram mappings here – SysML doesn’t really have an anatomy of diagrams and content or document modeling concept.
- The semantics of Views and Viewpoints is very vague and weak.
- Use of Views and Viewpoints
- It would be nice to show the transitive result of certain relationships. for example, if A is a subclass of B and B a subclass of C, then it would be nice to show an (inferred) generalization relationship between A and C when it is useful; the same observation applies to compositing

associations.

- Useful to communicate basic model structure, but otherwise pretty superfluous.
- View/viewpoint not sufficient; need non-owning organizational constructs, standardized automatic view construction, possibly profile diagrams
- No defined semantics, little value in comparison to effort to produce
- Steep learning curve for the tool and difficulty dealing with complex projects (visual clutter.)
- possible usage

Parametric diagram

- Should provide a more graphic interface like Labview
- Didn’t found yet how to use them in an appropriate way, adding value and understanding to the model.
- Access to correct values declared in bdd
- This is an underutilized, and could be more closely coupled with the ports to which it maps
- Difficult to use and associate with other model objects
- Parameter interaction typically falls into two categories ... too simple or too complex to be modeled with an equation
- Training - purpose of parametric diagrams not clear to beginners.
- Struggled with trying to understand how to apply this diagram to our problems. Use of these diagrams seems to be more associated with an approach or a way of thinking while conceptualizing system design solutions.
- If they are not executable, they are useless and a waste of valuable time. I know F=ma, I don't need to model it. But if I have an executable model, then parametric diagrams are useful.
- Not really sure the best way to use these diagrams
- Limited experience with this one, yet, but seems like each project needs to build your own library. Would be useful if tools had an packaged library to get started.
- Takes too much effort to first define an equation then create a usage and then create instances for everything
- Lack of precise semantic definition - need better language such as Modelica
- Formal link between the constraint parameters and the arguments of the opaque expression
- Behavior parameter can't be used. Requirement attributes can't be used.
- We believe that Parametric Diagram will become more and more useful as tools will start to offer support for their execution. We would like to try and define a standard constraint language in SysML 2.0 that will be intuitive and simple enough for the specification of mathematical constraints. One technicality that we reported for RTF 1.3 is the need to have an instance reference to bind a constraint parameter to a block attribute in a context of a part hierarchy.
- The use of nested constraints is unclear.

- See everything IBD, keeping these 2 separate is painful - imagine a separate BDD for blocks and constraints - please consolidate with IBDs no use for separation, no strong equation rendering interface like MathML or equation editor (not to be confused with tool issues – need some general interface of what and how can be used), can't explicitly map between parameters and variables in equations – there isn't a clean standard interface to describe how this can be done, can't explicitly differentiate between a-causal equations and equations known to be causal, cant retain nice mathematical expression separately from various underlying representations for script calls or other solver notation, .
- It is not clear why a diagram different from an IBD is needed. They show practically the same things. / People have difficulties to see the difference between behavior described with activities and dynamic aspects of the system using parametrics, which act on static system properties. / Properties of Actions and Activities difficult to integrate, e.g. duration of an action could be bound to parameter
- Not mature - both concept and our organization to use it
- Confusion over which compartment equations belonged in - tool not following the SysML spec
- Concepts of value type etc too confusing. no clear understanding of how to apply complex math structures like vectors, matrices. Correspondence with other aspects of the model unclear e.g. causal description using seq. or act.
- Can get very large and hard to follow for complex relationships with many parameters. Good to have an unambiguous spec for defining equations.
- No input; didn't have a chance to use it yet
- Very powerful and very complicated. We've struggled with these both within our modeling team and with external stakeholders.
- It seems like real world mathematical concepts are much more complex that the simple examples I have seen. Not sure how to really use these diagrams or if they are useful.
- Suggest optional directionality to equation parameters, to make equations/constraint blocks reusable we need to ability to scale model values (to convert units) on binding connectors and between pre-existing equations
- No defined semantics, little value in comparison to effort to produce
- Steep learning curve for the tool and difficulty dealing with complex projects (visual clutter.)
- how these diagrams link to the others needs further explanation
- Binding to tag values and part attributes
- Add direction arrow on parametric parameters when helpful
- No issues but usefulness has not been conveyed adequately

Requirement diagram

- Just flat text, no mark-up
- Wouldn't manage the requirements in SysML. Density of information possible in boxes is unmanageable. Need to establish and promote interface to Reqmts Mgt tools.
- Needs tracing to rationale AND constraints
- Clear examples of linkage to DOORs: 1 extensively used. Visualization issues with too many requirements per diagram.: 1 Messy: 1 not useful: 1

- I would like to see the SysML specification be furthered to standardize the way these diagrams are converted to requirements specifications. Also, there is reluctance to use this feature in favor of dedicated commercial software products, partly in my opinion, because it seems too unstructured.
- Need to tell users that Requirements Diagrams are only for looking at small things
- Numbering.
- How to use it for requirements management?
- Really, only use these as throw away diagrams. Just use to create the relationship between a requirement and a block/ activity/ use case/ actor/ etc, then objects from diagram. Consider just drag and drop requirements onto objects without adding requirements to diagram itself.
- We need properties for requirements so that we can refer to these numerical values explicitly in other diagrams
- Cannot be used to manage large quantities of requirements. Good for documenting relationships between a few reqs.
- Only text-base requirements are supported
- Subrequirements need to be ordered. DeriveReq relation direction is confusing. Containment relation semantics is unclear (is parent requirement a normal requirement or used just for grouping?).
- Having requirements in SysML and some basic relationships such as satisfy in the SysML language is a basic need of the language, however drawing the requirements on a requirements diagram is not scalable and it seems that users prefer to use tables or connect to requirement management tools (non-SysML)
- Requirements don't allow structured text to reference other model elements.
- Amount of requirements for non-trivial systems quickly is too much to put in graphical notation - need much more powerful tabular views
- Text strings in a model are limited - need to express things more like constraints e.g. specifying the units and values of parameters in requirements.
- Requirements have no attributes and can therefore not be bound to parameter. This is very important for a seamless model.
- Requirements diagram does not scale up. Different to manage requirements over time in SysML tool.
- Handled requirement traceability in non-SysML tool, too many requirements to handle as objects on diagrams.
- Too flat. Need to be able to parameterize.
- Need a way to connect parameter to statement.
- Can grow large making it difficult to view and navigate.
- No input; didn't have a chance to use it yet
- Dependency arrow direction is not always obvious, but otherwise req diagrams are pretty straightforward.
- Move to UML; add ability to use templates for different type of requirements; requirements need parameters that are usable on parametric diagrams and elsewhere. use of namespace containment is problematic
- Hard to follow when there is a large number of requirements
- No defined semantics, little value in comparison to effort to produce

- Steep learning curve for the tool and difficulty dealing with complex projects (visual clutter.)
- Creating the requirement diagram as a tree is cumbersome and takes up lots of space.
- The direction of the arrowhead in a derived relationship is counter intuitive.
- association of interface requirements with interface classes
- linkage of DOORS requirements to the model is valuable
- Great in theory, too busy in practice. Visual representation of requirement allocation has minimal value.
- Cannot effectively support large numbers of requirements
- The tool doesn't provide an easy way to compile all requirements in a consolidated view. It is hard to keep track what has been defined, and what is not.
- Used extensively, no issues.
- Didn't find the generation of requirement diagrams needed much. Imported Reqmts from DOORS, created relationships and then removed Reqmts from diagrams/views.
- I tended to put requirements on the use case diagrams instead of making requirements diagrams

Sequence diagram

- Multiple threads are very difficult to model and visualize, yet they are often important
- They become easily very large, without providing a complete overview (just represents one possible scenario).
- better support for multi-page diagrams; connect notes to messages, states
- Artisan Capture of Object methods and attributes good. Animation really woke up customer
- These are not scalar
- Distinction between "send" vs. "call" messages not generally understood: 1 Modeling event driven systems that don't execute in the same sequence in each use: 1 none: 1 Overuse of UML 2.0 additions like loops and decision points. Allows for too many paths per diagram if not managed.: 1 Using them to show how attributes or values are updated: 1
- In Enterprise Architect, this seemed like a difficult diagram to use
- These should be the primary diagram for System Test. No one is really pushing this.
- Lack of correlation of information provided on the activity diagram vs. sequence diagram. The active region on the sequence diagram could be the same as the activity on another diagram; the message passed on a sequence diagram could be the same as the object flow on an activity diagram (diagrams could be complimentary views, but not treated as such)
- For the kind of systems I model they are not very useful -- maybe more relevant for software
- Difficulty in expressing periodic communications
- As it seems, the sequence diagram has not been expanded to include concepts for flow properties in SysML
- No SysML-related issues
- Can't expand/collapse groups of lifelines to switch blackbox to whitebox views (more @ Q60)
- Our users use sequence diagrams to model interaction between parts, discover interfaces of blocks, model validation (animated sequence diagrams) and testing (seeing if a model execution conforms to a set of sequence diagrams). We don't see any special need to extend sequence diagrams.
- The interactions couldn't show item flow, but only message exchange.

- Very confusing in a systems context. same goes for methods/operations. Activities and timing diagrams are much more desirable here for true system models. Problem with Sequence diagrams is they don't belong in SysML or at the vary least should be much more inter-changeable with activities.
- If a Call-Behavior of an Action is a Sequence diagram it is not at all clear how object flows and control flows are handled in the Sequence diagram. / Sometimes it would be useful to use allocated parts in sequence diagrams, e.g. when SW is allocated to HW one would like to use also the SW components in the sequence diagram.
- A diagram many people are used to use, but there is little place for it from a systems perspective
- Creating and destroying lifelines
- Correspondence between seq and act unclear but can be used to achieve same intent.
- No issues; pretty well defined
- We attempted to use these to represent control flow in a non-object oriented system, and that involved some SysML compromises.
- Linkage between AD, BD, SD, UCD and requirements was minimal which allowed them to become inconsistent
- We have not made significant use of Sequence diagrams, though we are considering using auto-generated sequence diagrams from activity diagrams as validation artifacts.
- System Architecture level diagram behaves the same way as in UML level Sequence Diagram.
- Give examples of activities as lifelines; needs to support continuous/streaming flows; can a flow be lifeline
- No defined semantics, little value in comparison to effort to produce
- Steep learning curve for the tool and difficulty dealing with complex projects (visual clutter.)
- What are the semantics of a Create() call on a block in sequence diagram? In software this is usually a constructor call. Is this the main classifier operation for a block?
- for analysis improve usage of message or classes from interface classes definitions
- Similar to UML
- Used extensively, no issues.
- Explaining the difference between a SW invocation versus the traditional SE data flow. In fact the Rhapsody tool finally just added a dataflow to the sequence diagram.
- Can not automatically specify flows as Primitive Operation Arguments - no drop down list Use of Flows
- Would like to see these applied to Patterns applications, along with associated Activity and other appropriate diagrams.
- Mapping flows to interfaces

State machine diagram

- Stat machine diagrams are ambiguous. It would be nice if a subset of the state charts was formal enough for a model checker or code generator (race conditions, two events that happen at the same time, ...)
- Overall good, but there was some issues where the state machine had to be flattened in order for events to be responded to in Activity Diagram
- Very good to describe the behavior of model elements.

- Portugal
- Okay at least defines need for
- States are underdetermined in dynamical systems where all states are pseudo-states
- Compilation was a challenge to run the state machine: 1 none: 1 not heavily used.: 1 Not used on our program: 1
- In Enterprise Architect, this seemed like a difficult diagram to use
- SAFE FWS
- Every system I've built has states and modes. We need to push state machines instead of activity diagrams for state/modes
- Minimal experience so far.
- Difficult to deal with nested states; the semantics for tying transitions to operations are vague
- No standardized formal semantics for guards and conditions
- No SysML-related issues
- State machines are the most commonly used diagram to specify behavior in Rhapsody since they are well suited for describing reactive behavior and because of Rhapsody's support for their execution.
- The protocol state machine is missing in SysML.
- Can't associate activities with state event transitions, can't use SysML constraints.
- Tends to get too software centric if conditions and events are too be used to enable simulation
- Use of pseudo-states (esp. entry and exit points, terminate, and history). Use of junction vs choice
- The notation for transitions is really awful. Matlab Stateflow Toolbox is significantly better.
- Please add way to explicitly parameterize states.
- lack of integration with activity diagrams such as the ability to pass parameters to a do/activity
- No issues; pretty well defined
- Utilization of the SysML flow ports together with the Rhapsody simulation code generator for the statemachines represents a significant challenge and can end up making the simulation implementation less transparent to the state chart observer.
- states do not exist as separate model entities, only as elements in a diagram (in Rhapsody)
- tie parameters/arguments to state behaviors, need ability to have state variables; should support bdd format for states so that BDD-internal paradigm can be repeated for blocks, activities, constraints, and states
- This have strongly defined semantics and actual can model certain types of computation
- Steep learning curve for the tool and difficulty dealing with complex projects (visual clutter.)
- Difficult to discern what to put for the effect in the [guard/effect] construct on a transition.
- Needs more explanation
- Did not use since I did not get time to determine how to apply them and I also have no background in their applications, details. We use Scenarios to capture the states and then the other diagrams for the states.

Use case diagram

- No issues. Generally intuitive.

- It would be nice to define forms to fill out for the textual part of the use case. It would be nice to have a way link the name of a block to a word in a use case. The block name changes, the text in the use case get updated.
- Modelers have difficulty with modeling the appropriate abstraction level and dependencies, esp. when representing a behavior (activity diagrams) associated with a Use Case, behaviors tend to overlap Use Cases
- Useful during the first analysis steps.
- Capturing and connecting ucd at different levels of hierarchy: 1 largely redundant with package diagrams, bdd's. : 1 No issues. Use Case diagrams were used. Use Case text descriptions are key.: 1 used little: 1
- Users are still using use cases to describe HOW and not WHAT. They are also using them to decompose the system, not the use cases. Training is necessary.
- Actually, we use this more as an index of scenarios, with composite links to corresponding activity diagrams. Find Use Case to Activity diagram links/navigation to be MOST useful. We're also rather unconventional about adding Use Case objects to activity diagrams - for several purposes.
- No SysML-related issues
- Use Cases are regarded by our methodology as being the "chapters" of the systems design functionality. These diagrams are highly used by our customers however sometimes people over-specify use-cases, we believe that some guidance on the usage of use cases could be provided in the standard in the context of systems engineering.
- The actor is a black box element. In Systems Engineering it is often necessary to decompose actors.
- Being able to represent ALL the stakeholders
- System boundary seems out of place. Too much like software model. Should just be block(s) or part(s)
- No issues; pretty well defined
- It has been challenging to build use cases and drive physical change to an existing complex system.
- We used rational rose and the tool wasn't the most intuitive to the team
- Linkage between AD, BD, SD, UCD and requirements was minimal which allowed them to become inconsistent
- No issues, generally well understood and simple.
- Add pre-requisite relationship among use cases, i.e., this uc must be performed before that one.
- No defined semantics, little value in comparison to effort to produce
- Steep learning curve for the tool and difficulty dealing with complex projects (visual clutter.)
- Out of date
- association of requirements to use cases and combining in a useful report
- Basic high level diagram - quite useful
- Getting the use of these to be either Black Box or White Box use cases
- Used extensively, no issues.

Other features (e.g., allocations)

- What about hierarchy, and time-based periodic systems. Should the language provide better support for this?
- There are inconsistent implementation of swimlanes in tools this can be a problem
- Consistency not always easy to be checked.
- Allocations and requirement relationships are critical for guiding model development.: 1 Export tools (e.g., Reporter Plus with Rhapsody), Interface to DOORs poor with Rhapsody, : 1 Relationship tables and matrix: 1
- CORBAS
- SysML is a language, but we need to spend some serious effort on suggesting workflows. The workflows need to reflect the available tools.
- Traceability.
- Context diagram should acquire more importance, and perhaps allow to define syntax is and semantics for interoperation with other systems external to the model represented in it.
- Lack of effective tabular views. Most vendors are so focused on diagrams; they fail to realize that sometimes working in tables is easier/faster than the diagram. Need more tabular perspectives!! with ability to manipulate/ add columns/properties to the table.
- Standard ports - problems with bi-directional messages. concept is foreign to system engineering. lack of support of data broadcasts through standard and flow ports. difficulty modeling physical flows.
- Allocations should not generate dependencies between allocated elements
- No integration of different model parts (e.g. conveyed info on sd msgs vs. item flows on connectors in ibd) Q60
- No explicit support for variant modeling (e.g. for product line engineering)
- Being able to easily depict decomposition in a tree format
- One general issue the way things can be combined on diagrams. Much of which I have described above, however the general issue is that it is desirable to mix structure, constraints, behavior and states in diagrams. Also, there is no way to make 2 blocks or parts touch and specify the nature of that relationship (like a stack diagram), no Editable Table versions of diagrams possible, editable text of model is only XMI - too complicated - need human readable text-version of SysML. Also I want to incorporate graphs, plots and charts as part of my unified spec – not generate them per se just link them perhaps the way I would with an equation in a constraint and I can't.
- <<allocation>> should not be a dependency as the direction of the dependency relationship is not always clear, e.g. when deploying software blocks to hardware blocks the software does not really depend on the hardware nor the other way round.
- The usage of <<allocation>> in definition->definition and usage->usage is not clear but this might be a methodology issue. The spec must be clearer about allocation schemes of usage->usage, definition->definition Currently the swimlane can also represent a block but when an action w/ a call-behavior is allocated what should happen? How should the action and call-behavior be allocated correctly? as an action needs the context of an activity.

- Allocation ObjectFlow to ItemFlow: The ObjectFlow (Edge) describes that in the context of an Activity the output of one Action is bound to the input of another action. In the context of a block a item flow describes the flow of an object from one part or port to the connected part or port. The allocation of the ObjectFlow to an ItemFlow defines which ObjectFlow corresponds to which ItemFlow in a given context. Supplier and producer and context need always be defined.
- Allocation strategies depend very much on the method used. The difficult part is to use them properly, e.g. the chain: Use Case -> Behavior -> Structure / Allocation is a stereotype of UML abstraction and the semantics (i.e. the exact mapping) of allocate are not defined in SysML.
- Allocation Pin to Port not addressed in SysML standard 1.1
- Allocations work fine in principle, but are not properly implemented in the tool we use.
- I don't see a way in SysML to easily organize levels of elaboration or detail that develop from one diagram
- In general, SysML diagrams are "locked" in terms of not being able to show the "result" of some inference we can make about the SysML model. Whether such inferences are made by, e.g., OCL query engines or by the user are beside the point. There should be a generic support for some kind of "derived" content originating from a particular source -- e.g., OCL, human, reviewer, ... -- and one should be able to indicate how the "derived" content relates to the model content (e.g., transitive generalization; transitive association, ...)
- I still need to create text base reports. I would like to see more requirements for tools to provide specification, interface design documents, etc. You can put lots of great stuff in the model but its tough to get anything out of it.
- Need state machines for interfaces; probably need an instance-like solution
- Steep learning curve for the tool and difficulty dealing with complex projects (visual clutter.)
- Not an easy way to provide a trace between system states & modes, requirements and use cases. Allocations help but it's tedious.
- When dealing with interfaces (especially external interfaces) at progressively more detail (IBDs), it would be beneficial to have some type of interface "aggregation" representation (e.g. a single external interface represented on a given IBD is comprised of many external interfaces at the next level detail IBD(s)).
- Requirement dependency relationships; allocation relationships
- allocated to vs. allocated from takes some getting used to
- Did not apply to the enterprise conceptual model.
- Ability to show vs. hide Ports was very useful

Question 6: What part(s) of SysML were hardest for your stakeholders to understand?

Open Ended Responses

- Activity and Parametric
- That the SysML represents a formalism, not just blocks and arrows.
- The problems we encountered are not due to SysML or UML per se, but rather, are due to a general knowledge deficit regarding the use of modeling (particularly the interrelationships). Also (and this is big), people have a hard time with abstractions in general, and with the abstract notion of a Repository vs. a Presentation (a document). Our folks live in a document centric world, and that is a hard habit to break.
- Package Diagrams did not seem to add value. Sequence diagrams are interesting, but a person almost needs notes attached through the sequence diagram to explain what is going on. It would be nice to standardize a way to document the sequence diagram / activity diagram and to tie the text to the use case it is derived from. There should be an inherit symbol that goes in the opposite direction of the generalize symbol.
- Internal Block Diagram
- Parametrics
- Block definition Diagrams to define blocks (associations, compositions, aggregations, inheritages,...)
- Activity diagrams and the interrelation between all behavioral elements in general.
- Sequence diagrams
- State machine, ibd
- Stakeholders were able to understand clearly all the diagrams after a little explanation
- Not to proliferate new objects versus using previously defined objects in sequence diagram. Gap between SW requirements and Sequence Diagram extraction was awkward
- How the models relate to the real world.
- Block Definition Diagram
- Block diagrams, IBDs and formulated rqmts from ADs and SDs: 1 Parametrics, "call" vs. "send" messages, generalization.: 1 ucd, bdd, ports - Benefits of using SysML: 1 We mostly used common UML/SysML diags... culture change from text to diagrams was most difficult for some stakeholders: 1
- You're assuming that stakeholders WANT to understand SysML. In my organization, they don't. I have software engineers who make statements like "code is self-documenting ... requirements are unnecessary". Such an organization needs effective, plain-language tutorials to persuade them that modeling with SysML is going to help them achieve their goals. These do not yet exist.
- Sequence Diagrams
- Portions of diagrams when judiciously extracted and explained are easy to understand. “Is-a” vs. “has-a” relationships, when unexplained however imply that the user stakeholders should be trained or at least indoctrinated into UML/SysML which in my experience is a bit of a hurdle.
- SysML

- Flow ports. I see them overused. Many customers are trying to use flow ports to model computer bus messages with the full DESIGN SPECIFICATION (ICD) in the flow port/flow spec. Swim Lanes. I see many Systems Engineers modeling actors because they can assign a swim lane to represent an actor. I don't think that SysML should be changed, but I think we need to train users better. Activity diagrams instead of state machine diagrams. I see far too many clients who use activity diagrams when they should be using state machine diagrams. The activity diagrams have event acceptors that cause the transition from one action to another.
- Multiplicities on relationships, especially composition relationships. This is a concept borrowed from UML, I know, but I had problems describing how different instantiations of the same block could have different links present.
- Use of Internal Block Diagram and port definition. It's not easy to link the IBDs to the other diagram.
- Parametrics
- None, the diagrammatic representations are straight forward and easy to understand.
- Parametrics Diagrams
- IBD with interfaces vs. activity diagrams & its flows
- Difference between BDD and IBD. They confuse this often.
- package, state machine
- That the model is more important than the diagrams -- there is too much emphasis on making the diagrams look "right" and pretty, without asking questions about the meaning and how this meaning should be captured in the underlying abstract syntax. To do this well, significant understanding of the abstract syntax of not only SysML but UML is required, and this is imposes quite a steep initial learning curve.
- Inheritance, standard ports.
- The "token based" semantics of activity graphs is powerful but not easy to understand. Interactions seem easier but they are misleading because the "trace based" semantics is very tricky to use properly and is not convenient for complete specification of the behavior.
- The port concept and related concepts caused most confusion ("What is the difference of a port to an interface? What is the flow specification about ?")
- All parts of SysML are understandable if the language was introduced in a training before.
- Combining software-hardware (or logical-physical) models initial, default, context specific values requirements grouping and deriveReq relation relations between block and behavior definitions subsystem or domain definition. artificial "context" definition to be able to use IBD
- Not observed yet. To date, have exposed stakeholders mainly to IBDs as context diagrams and sequence diagrams as scenarios. In both cases the notation is sufficiently familiar and intuitive that there hasn't been any sense of confusion on their part. Plan to expose them to more types of diagrams including requirement, state machine, use case, and BDD in coming weeks (after RFI deadline).
- 1.Execution semantics of activity diagrams
- 2.How to perform functional decomposition in SysML?
- 3.Modeling continuous behavior and data flows
- 4.General methodological issues: SysML perceived in general as too complex
- Internal Block Diagrams.

- For stakeholders without UML background hardest to understand were direction of arrows for relationships (e.g. <<trace>>, <<satisfy>>) and open/closed diamonds (shared or composite aggregation) between elements
- Symbolic notation in the grammar (too often this is ambiguous)
- The meaning of all the different diagrams and the rules for them e.g. you cant put activities on IBDs or a state event transition, the lack of familiar concepts (like just putting values on things – Systems engineers decorate with numbers – and notes or UML instance diagrams just aren't good enough, stereotypes and how they are used is confusing as well.
- Block diagrams BDD, IBD if the style is not consistent. There are sometimes so many ways to display parts, stereotypes, compartments, references, structure, allocations, ports, interfaces, etc. that if no a rigorous style is applied the reader gets confused. In particular people who have never seen SysML IBDs before (like industry contractors) but have drawn block diagrams in their own syntax and semantic have often difficulties with the level of detail or the style. The same information can look quite differently and people are overwhelmed which line or arrow is important in a certain diagram. Often the direction of the arrows is confusing for people who have NO software or UML background. They consistently would draw arrows (like associations, allocations, dependencies, realizations, etc.) in the opposite direction. Interfaces vs. ports The interfaces are called interfaces but are in fact only service oriented interfaces from UML. To system engineers this does not make sense. They have optical, mechanical, electrical, etc. interfaces which are in fact the standard ports. The term interface for this particular usage (i.e. realize a service interface) is quite confusing. Terminology People get confused by all the terminology like part properties, value properties, shared properties, etc. And sometimes there is more than one term for the same thing, e.g. shared property, reference or aggregation.
- Activities - there is a LOT of confusion about Pins, Parameters, and Parameter nodes. Also because for example <<rate>> is not available for all of them.
- In general there is too much freedom for the modeler to make invalid model connections. E.g., in the tools we use we can have flows connected directly to blocks or parts works fine. From a users perspective the question is what is valid and what are the semantic differences? Also, I can have an object flow connected to a pin in one and directly to an activity on the other end. What does it mean - if it is legal
- relation between ibd and bdd; relation between pin, parameter node, and parameter; relation between constraints on blocks and constraints in parametrics; weakest elements so far are views and viewports;
- The fact that SysML elements should always be mapped to elements in the domain-specific models which they are familiar with.
- The pairing of static and dynamic ways of looking at a system--use case and activity, for example--is not discussed as such in the documentation I have seen, but it's one of the strengths of SysML ... and I think stakeholders were perplexed because they assumed that every kind of diagram would communicate in the same way, while in reality SysML gives you both static and dynamic ways of looking, so you must know to switch
- 1) Concepts of "definition" and "usage" 2) Relationships between behavior described within Use Case, Activity, Sequence, & Internal Block (as shown via ports/interfaces & operations) Diagrams. 3) Use of synchronous vs. asynchronous messages 4) Misuse of relationships between use cases (includes, extends), i.e., the hierarchical use cases as functional decomposition problem.
- See answers to #5.

- Between me and my colleagues, to date we have taught SysML in multiple hands-on short courses for multiple industry/government organizations (over 100 participants total so far), and also 3 offerings of a full semester academic course (with ~50 students total combined). These courses include executable parametrics and activities. We have also been part of the OMG SysML RTF and continued research in MBSE that includes a heavy usage of SysML and related tools and technology (see http://www.pslm.gatech.edu/topics/SysML/ and http://www.buzztoys.gatech.edu/), including being part of the INCOSE MBSE Challenge (http://www.pslm.gatech.edu/projects/incose-mbse-msi/). Thus, the feedback in this overall RFI response in general is with the above experiences in mind. Unfortunately, due to other constraints this RFI response is sparse and gives just a few highlights. Hard to understand aspects: a. Part properties vs. reference/shared properties. Similarly for instances vs. usages. b. Parametrics (until they see and try several working examples that solve). c. Confusion on when to use what (e.g. the different types of ports for ibds). d. How to get the various constructs to work together (e.g. a single problem that uses activities, state machines, and parametrics all together). e. Being overwhelmed at first by the many different choices.
- Sub diagrams like Parametric and Internal Block diagrams
- Activity diagrams: they look simple but few people get them right. Hooking SysML activity diagrams to fUML requires a lot of modeling investment few people are prepared to make; this could be addressed with some practitioner-oriented profiles extending fUML. State machine diagrams:
- I don't know about my stakeholders, but I find the lack of support for instances puzzling.
- BDDs, generally a lack of support for alternate displays using graphics etc.
- parametric diagrams, direction of requirements relationships
- Semantics of the diagram elements, when elements look similar, such as open/closed arrow heads, diamonds, etc.
- Within the artifacts we created, the most difficult to understand was the actual OMG Standard since it builds on UML. If a stakeholders doesn't have the knowledge of UML then he/she wouldn't understand the standard. I found that presentations such as OMG Systems Modeling Language by Sanford Friedenthal from INCOSE 2006 Conference in Orlando provided a clearer guide.
- Use cases.
- Translating SysML to embedded systems
- Our implementation made it hard to illustrate different instances and how they relate to one another.
- definition and uses- constructs of the lower levels are always desired first without effort allocated to top level definition. Use case diagrams are not understood and efforts produced for products are similar to poor functional allocations using physical components
- Linkage between the UML artifacts and requirements
- Levels of abstraction of the system and the delegation of behaviors and interfaces between those levels of abstraction.
- BDDs: People assumed them to be schematics. They also had trouble with Composition vs. Aggregation relationships Parametric Diagrams are alien to most people.
- When applying SysML to a new domain its very difficult to figure out how to use. We have spent a lot of time just trying to figure out how to use SysML to represent data on an interface (such as MIL-STD-1553B).
- Associations

- ROI on initial incorporation of MBSD at a BU. Until reuse and reduced rework can be quantified, MBSD is a hard sell.
- Parametric Diagrams
- The inherent integration of the model in the supporting tool - in this case System Architect.
- Advanced act/seq diagrams are not obvious of course; confusion between reference and depletable store, how to show specific configurations/instances; requirement relationship directions
- BDD's are really difficult for folks to get the concept of. Relation of use cases to activity diagrams. Understanding that parametric diagrams do NOT calculate anything and are just representations of parametric relationships. Ports (both standard and flow), flows, flow specifications on IBDs cause a lot of confusion.
- The relationship between diagrams
- BDD and IBD: use of ports and interfaces Activity diagram: use of tokens, {stream}, <<continuous>>, and <<discrete>> Frames: naming of
- Very few stakeholders understand any part of SysML
- Unless they have been involved from the start, there is always a lack of confidence the model and systems are compatible.
- Aggregations vs. directed compositions in block diagrams.
- How to interrelate the various diagrams/blocks for decomposition and analysis. (how activities, use cases, requirements, IBD are used together in a meaningful method)
- parametric diagrams
- Ports; Levels of abstraction: Specifying physical and logical ports, parametric diagrams: Value types
- Still too new a process to evaluate. The UML construct in a mechanical engineering world is still difficult to sell, in my view.
- The relationship between a bdd and a particular ibd, the suggested synchronization between diagram artifacts and requirements (refine vs. satisfy). Parametrics needs to be introduced with more examples showing usefulness.
- Ports, flows, flow items, interfaces and contracts
- Requirements diagrams - outside of DOORs
- The flow and tracing of requirements.
- Formal meaning of notations - too much ppt engineering and not formal definition of how to diagram SE concepts
- Sequence of events flowing through different parts of system.
- Package diagrams, requirement diagrams
- Small project with no external stakeholders, but selling model based system dev was very difficult. Resistance to new methods of sys eng. Project terminated prematurely due to program cancellation.
- The value of the systems engineering process as a whole.
- Parametric diagrams because they weren't used to seeing diagrams in that format.
- People to hardware to processes especially in a service type environment when the services do not all belong to the system under discussion. This may be more of a poor presentation but internal versus external is becoming fuzzy

- Operational process model
- Sharing of control to BDD elements when multiple distributed team members needed access to them.
- Ports and interfaces
- How to map Flow Ports to Standard ports for System to Software/Hardware Interface Definition.
- Did not get far enough to give the diagrams to the stakeholders
- Ports/interfaces/flows/linkage to flows on sequence diagrams/mapping to requirements and message classes
- Have no experience of SysML

Question 7: If you defined your own stereotypes, please list the stereotypes and the model elements they were applied to.

Open Ended Responses

- <<agent>> Extends actor that works within a system boundary, interacts with other agents and exhibits intelligence. <<Intelligent>>Extends UMLBehavioralClassifier and represents hardcoded rules thru weak artificial intelligence <<context>> Extends ContextDiagram plus ExecutionEnv from UML L3, thus the SysML Profile is more complete than traditional L2 extended UML. <<spacetime>>Substantially different from timing diagrams, this <<context>> element exhibits both time and frequency domains. All these extensions support Model Based Simulation directly from UML models.
- For use in modeling enterprises and business process: <<document>> = a physical presentation of a space in the repository <<analysis>> = an external (from the model) application of logic <<logical device>> = a partition of a subsystem, but not viewed in the physical realm.
- Test Stereotype - Similar to Requirements
- Tend to be domain specific.
- Stereotype "store" applied to blocks (and creating parts) to model memory regions of the system(with input/output flows).
- system, subsystem, external system: block fragment, secondary: use case
- <<agent>> An actor that exhibits intelligence (AI, or hardcoded) working within a system. <<intelligence>> a classifier. <<context>> Temporally extended classifier
- Not Applicable
- <<logical>> & <<physical>> blocks: 1 MOSA levels for hierarchy – blocks, use cases: 1 Safety tags - Requirements diag and Sequence diag elements, : 1
- <External System>, <System Function>, and <System Subfunction> stereotypes for Blocks and Parts
- Toolcorba
- SOA services get applied to operations. SOA Clients and SOA Providers applied to Blocks.
- We defined several stereotypes to restrict communication access between various blocks. Our model (TEAMS) was intended to be used as a basis for another model (probably also in UML/SysML, but not necessarily). Associations in the secondary model could be added from the first, but only if it did not violate certain rules. These rules were encoded with stereotypes on the blocks.
- Documentation to be traced to.
- Stereotypes for port types and connector types. Examples include Ethernet or optical ports and connectors.
- Have not defined my own stereotypes, but would like to define a stereotype for and even a Test Diagram
- Blocks: External, CSCI, HWCI, logical block, physical block
- interfaces - basically creating layers of class types for logical representations & physical representations
- Mechanical: system, lru (line replaceable unit), lrm (line replaceable module) electrical: processor, asic, fpga software: application, csci interfaces, logical: data item interfaces, software: data stream interfaces, electrical: data link interfaces, mechanical: mech link

- We developed stereotypes for continuous dynamics (Modelica) models and simulations we developed stereotypes for elementary functions and flows (as defined in the German systematic design school) we developed stereotypes for the application domains of hydraulics and space systems
- Used for requirements and relationships between them
- Blocks, FlowSpecifications, dependencies, comments, packages
- Software property - Part, used to model software parts of the system chain property - Part, used as logical encapsulation chain for functional combinations of hard and software system components branch - Dependency, used for modeling configuration branches deploy, Dependency, special kind of allocation used to show the software/hardware deployment referenceOf, Dependency, used as a connection between original and reference element. The reference element is just a reference of the original. It it similar to a hyperlink and a web page. The concept is used to avoid modeling of equal system parts in the architecture model.
- Requirements, blocks, parameters
- Various document stereotypes assigned to blocks, representing standard document artifacts (typically MS-Word) in traditional SE process. <<Applicable Document>>, <<Reference Document>> applied to relationships between blocks representing realworld documents. Various stereotypes to define elements of "controlled interfaces" = interface controlled via formal ICD (models concepts of controlled interfaces, services, message groups, messages) applied to blocks and ports <<data store>> to class <<glossary item>> to any element <<gold standard>> to any element Many others: <<hardware>>,<<software>>,<<physical>>,<<logical>>,<>,<<nicknamed>>, <<modeling standard>>, <<important design decision>>, <<important unknown>>, various others related specifically to remote sensing satellite mission domain
- As tool vendors we avoid adding new stereotypes to SysML
- SYSMOD Profile: (Activity: Continuous activity, Essential activity), (Actor: Actuator, Sensor, Boundary system, Environmental effect, External system, Stakeholder, User, User system), (Class: Document, Domain block, Functional block, External block, Parametric context, Subsystem, System, System context, Hardware block, Mechanical block, Software block, Feature, Business requirement, Constraint requirement, Extended requirement, Functional requirement, Legal requirement, Objective, Performance requirement, Physical requirement, Reliability requirement, Supportability requirement, Usability requirement, Variation parameter), (Association: Electrical association, Mechanical association, Software association), (Abstraction: Represents, WeightedSatisfy, WeightedVerify), (Interface: User interface), (Port: Junction port), (State: Mode), (Use Case: Continuous Use Case, System Use Case, Secondary Use Case, System Process), (Package: Variant, Variation), (Packageable Element: Variation Point)
- <<concept>> to <<block>> and <<activity>>. Also domain specific stereotypes for typical categories/classes of system elements like Ground, System, Subsystem, Activity, HardwareEquipment, SoftwareProduct, SoftwareModule, Propulsion to <<block>>
- Too many to list. Stereotypes are very confusing. This concept needs MUCH simplifying. For example it would be much more preferable to just use blocks and inheritance and avoid stereotypes completely. Most domain specific, but certainly interface, function, risk, trade, team, role, host of basic performance cost and schedule properties to name a few.

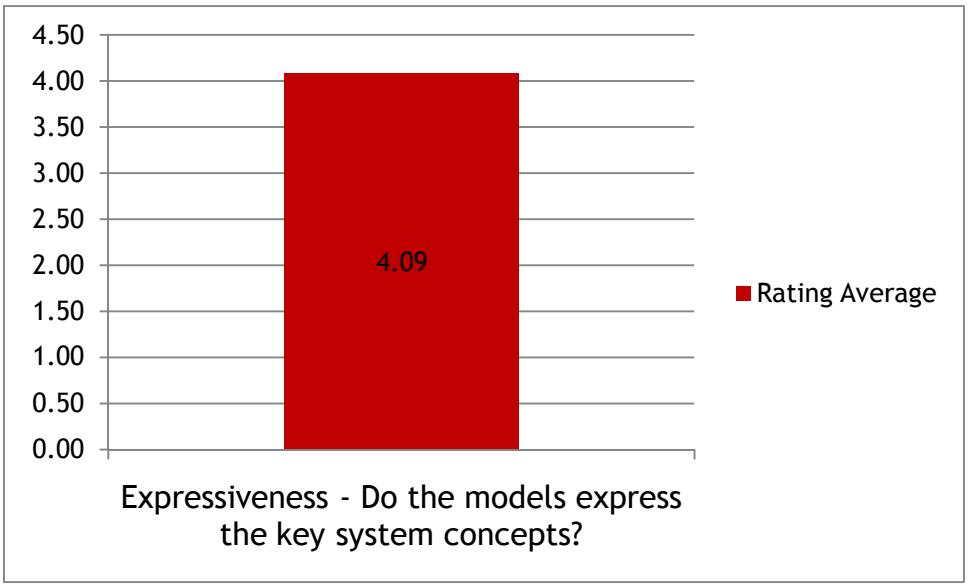
- Requirement (Stereotype specialization): Objective Connector: logical, information, physical, electrical, optical, thermal, icd Port: <<junction>> port for delegation port at borders of parts but diagrams get cluttered because there is no specific icon for it. Block: functional, physical, software, electronics, analysis context for container of parametric diagrams, <<portsgroup>> Comment: ERROR, Issue, parsed Pin: QoS which includes latency, jitter, clocked (later on we discovered that there is a UML profile for quality of service) Diagram: clocked
- Plenty of stereotypes for domain specific engineering, e.g., avionics.
- We haven't defined our own stereotypes yet, but we would like to define such stereotypes as functional element (block), physical none (block), communication protocol (block), logical interface (connector), physical link (connector), etc.
- Most of the metaclasses in fUML + SysML-specific metaclasses such as flow ports/properties/specifications/items.
- We have defined and are defining many stereotypes to represent a large number of concepts from system engineering and space flight, e.g., component, hardware component, flight hardware component, facility, risk, process, document, etc.
- Too many to list. Profiling is a powerful capability.
- System, logical, hardware, software
- Requirements Analysis stereotype applicable to Requirements
- We used stereotypes to define Rhapsody tags that supported information exchange between Rhapsody and other tools (interfaces, requirements) and that supported VBA scripts that automated design rule checking / compliance.
- Configuration IDs: <<606E>> <<606F>> <<606G>> applicable to blocks Data Protocols: <<RS422>> <<Power>> <<1553>> <<TTGbE>> applicable to connectors
- logical, physical : applied to blocks, ports, <mission type>, <segment type> applied to actions, <configuration release number> applied to blocks <subsystem type> applied to blocks, activity diagrams
- Subsystems and Components,
- Reuse, newDev - component, Block, Class
- special forms of problem (tbd, tbr, tbs, tba, action items); special forms of actors/use cases for threat analysis; stereotypes to distinguish between real world elements and system elements; context diagram
- Fw for firmware optional for optional branches of an activity diagram message block / message field to define blocks that are messages and the fields within that message
- logical, physical, firmware,
- <<aggregate>> - identifies a requirement derived from multiple sources.
- Functional Block - applied to general block. this allowed to structure the analysis of general system functionality without assignment. (highly abstracted view of a system)
- Physical, logical, locality, IO, HW (hardware), SW (software),
- I created the event stereotype to correspond to blocks that were used as events. I created the menu stereotype to indicate what blocks were created from menus the user identified runtime parameters on. I created the message stereotype to identify blocks that were used for external message IO.
- Creates, captures

- Baseline stereotypes (project definitions), Documentation stereotypes (report generation based on the model)
- We did define stereotypes that were applicable to our classified application
- Used stereotypes defined by others.
- Blocks could be stereotyped as System Segment, Element, CSCI, CSC, CSU Use Case Diagrams were stereotyped as Segment Functions, SEIT Threads/SubThreads, Development Threads/SubThreads
- We created sterotypes for our modeling process, which included systems, subsystems, and components

Question 8: Rate how well SysML supports the following language effectiveness measures. Please elaborate on your answer.

Results

Where 1=Low Effectiveness 3=Medium Effectiveness and 5=High Effectiveness



Open Ended Responses

- I am rating OMG SysMLv1.1. It is very expressive, but often requires profile extensions for more complete expression. The UML is more expressive and complete, yet still requires extension for spatio-temporal modeling of dynamical systems.
- The SysML language provides pretty good support for expressing concepts. I am not sure I could come up with a better language. The problem is not in the language, but rather, in the ability of people in the team to utilize abstract concepts and to see (or accept the existence of) interrelationships. For example, when defining relationships between classes (blocks), or when defining processes, the internal coupling of the system emerges. When it does, many people react by saying that the "model" is too complex. The reality is that, lacking a rigorous modeling method, these people would never have discovered this coupling until a disaster occurred, and would otherwise hope they can get away with it long enough to find their next job. The existence of a method for identifying such coupling destroys their formerly held illusion of independence. This causes brain-cramps. Motto: Ignorance is bliss. We were all happy until the existence of modeling tools shattered our illusions. Your next survey might be directed to answer the question of why some many people with the title "Systems Engineer" are so ignorant of "Systems Engineering". Sorry for the tirade.
- I have trouble expressing data dictionary information. This almost needs a table or a form. I have trouble expressing user interface data.
- Yes, the models are great. I do miss the Deployment and the component diagrams from UML.

- It highly depends on how precise a system has to be defined. It may become easily very confused.
- Far better than written specifications, but due to the lack of formal semantics people can still (mis)interpret the model?
- With Real Time constructs
- The limitations are often not in the language but its use.
- SysML is a useful adjunct to UML.
- A SysML model can bring all program systems artifacts together into a single coherent model. We have had great success.: 1 Most expressive systems language available.: 1 Really want the ability to use blocks as templates, Ex –describe connectors, type, num pins & what is flowing through it.: 1 SysML provides precision in diagramming. Drives completeness in modeling.: 1 SysML provides what the engineers need, the tools are not uniformly conforming to the standard: 1
- When used with the applicable methodology, SysML expresses key system concepts very effectively.
- Effectiveness of SysML diagrams is highly dependent on demonstrating how they can be used to define systems through domain specific examples.
- For an arguably thorough example of SysML as applied to development of Federation systems, see: Haley, T.; Friedenthal, S., "Assessing the Application of SysML to Systems of Systems Simulations" Proceedings of the System Interoperability Standards Organization (SISO) Systems Interoperability Workshop, Providence, 13 -18 April 2008. http://www.sisostds.org/index.php?tg=fileman&idx=get&id=2&gr=Y&path=Simulation+Interoperability+Workshops%2F2008+Spring+SIW%2F2008+Spring+SIW+Papers+and+Presentations&file=08S-SIW-100.pdf
- CRYPT32.DLL
- It isn't the language; it is the lack of a formal workflow (procedure) that guides the modeling. Too frequently I see clients who have no experience modeling trying to set the processes/procedures they will use. I also see procedures that do not reflect the tool that the team is using. This attempt to be "tool agnostic" ends up causing far more work (and frustration) for the end user.
- Composition relationships were easy to imagine from the SysML diagrams. IBDs were especially helpful in this since the boxes are drawn physically encapsulated by other boxes. Single blocks with multiple composition relationships led to confusion, though.
- The use of the Block Diagram in representing domain organization, enterprise organization and systems is outstanding. I do miss the concept of a "Deployment" diagram from UML
- I believe that SysML allows you to really dive into the system behavior and the activities/operations needed to accomplish different threads. The IBDs are useful to explain structure but for some it is hard to go through a set of diagrams if modeler decides to have multiple IBDs for different level of abstractions. Need better way to integrate lower level structures up to higher level of abstraction for those that one full view of system down to lower levels.
- The models are very easy & quick to work with for use cases & sequence diagrams but are cumbersome for capturing interface information that covers physical, electrical, & logical / physical characteristics (wondering about publicly available class libraries as an option)

- Getting there. Still has some major weaknesses that need to be addressed. #1 item in my book is associating blocks with connectors (lines). SysML does not treat lines on a drawing as real objects themselves, but only relationships. Oftentimes, we want to express something on a drawing with a line, that represents a real thing. We can see how to add an item flow to a connector, but tools will not allow us to type the line itself. See SysML std page 73, Figure 9.5. I see this example, but tools won't let me 'do it right'. I can type the flow on the line (Fuel), but not associate the line itself with a block (Fuel Supply Line). I've tried in both Sparx EA and IBM Rhapsody, and got lots of advice on a SysML forum, but no success!
- Yes, as long as you are willing to extend them with stereotypes when necessary
- The time concept is missing but MARTE can be used. We experienced problems with the allocations that generate unexpected dependencies. The problem was resolved in MARTE with <<assign>> allocations based on Comment rather than Abstraction.
- Am able to define system in terms of context (interface to external entities), requirements allocation and derivation, logical/functional decomposition, externally-visible behavior, scenarios to expose behavior at lower levels of decomposition, content of information flows, state behavior. Have yet to really explore activity modeling or parametrics.
- The language is quite expressive and the basic systems concepts are covered in our opinion (except perhaps distinct domains such as product lines engineering). Some users expressed an opinion that the language is too complicated or "too rich"
- Good SysML models give all stakeholders an overview of the key concepts and serves as the glue between different engineering disciplines.
- Key system concepts expressed, but weak on semantics and notation for nested more than one level deep whole-part decomposition and interfaces, e.g. block/part/port/connectors.
- Excellent, except too much latitude is given to the modeler resulting in too much variation. Each user will depict system diagrams in their own manner.
- This is a mixed bag. its great that SysML separates the concepts of structure and behavior(I consider parametrics behavior). time is missing however so any time-related concept must be hacked together with notes and explanations which is fairly poor. diagrams only sometimes allow the depiction of the overlaps of key systems concepts. For example showing an IBD with the different behaviors inside is useful – but not possible or well understood. Activity diagrams are behavioral, but allow parts to be represented by swimlanes. BDDs allow everything - activities, constraints, requirements etc. SysML also lacks a more formally central concept of information concepts. Parts and parameters are identified but what do they mean beyond just facilitating some local expression in the diagram. Blocks and stereotypes get the job done, but flow based properties, activity parameters and all the other disjointed concepts of information exchanges are hard to understand.
- A good model with well defined diagrams gives all stakeholder an overview of the key concepts of the system and serves as the glue between different involved development disciplines.
- SysML supports the key modeling concept we would like to see (and a few more), but all information is not suitable for SysMLification. This is foreseeable and we integrate lots of free text and graphics in the models.
- This largely depends on modeling guidelines and practices not on the language alone.
- SysML has all sorts of building blocks necessary for expressing how systems are (or should be) constructed (although you need to define a mapping from SysML constructs to elements of your domain model).

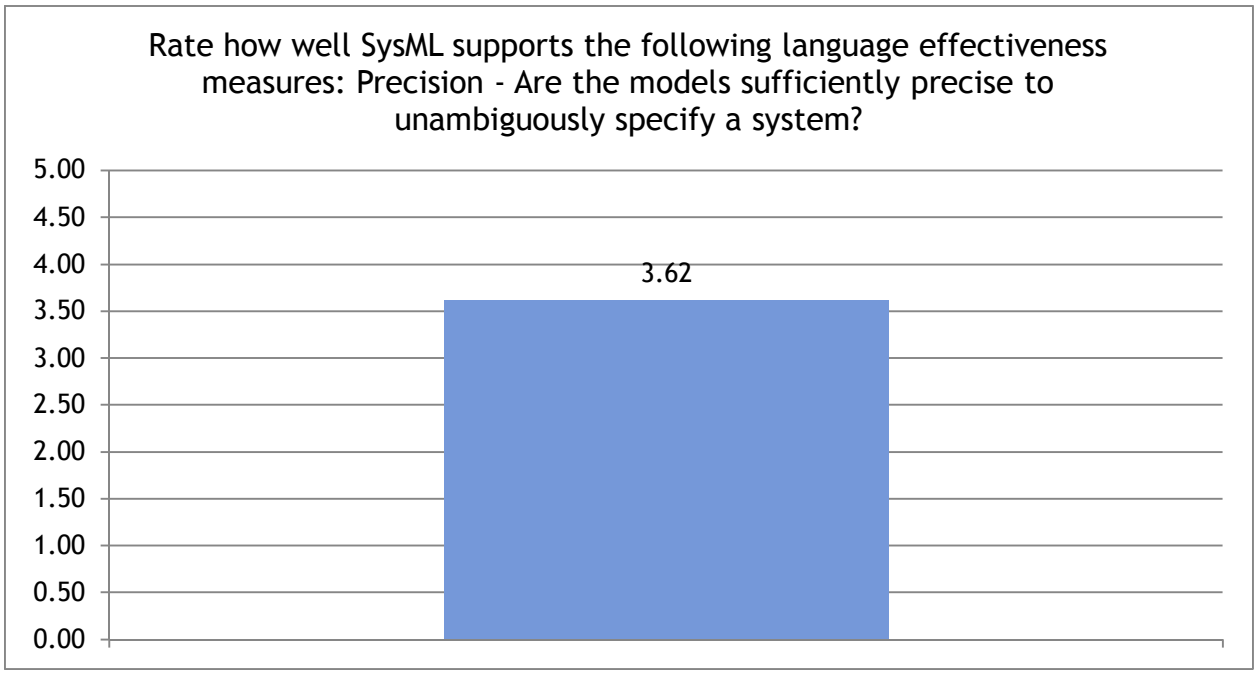
- Able to use SysML concepts to express key system concepts.
- The problem is that without a 1st class notion of view/viewpoint where we can relate the contents of two viewpoints; then it is difficult to make models that reflect more than one viewpoint. Also, it is difficult to properly distinguish: - a system vs. the environment the system is in - basic concepts of control system theory incl. the notion of system state; estimation/control; - the concept of 'objective' and how this relates to requirements and state control/estimation - the notion of 'time' -- is there a global clock for time or are there multiple clocks? If there are multiple clocks, how do they relate to one another?
- Yes. I think SysML is very expressive. the only thing absent is time related aspects. it would be very nice addition to give some more explicit concept of time or time representation to behavior diagrams and include uml timing diagrams.
- I believe mixing structure, behavior, parameters, and requirements is a powerful, ambitious goal.
- In general our models have expressed key system concepts. We have struggled to find an appropriate level for expressing system services (primarily messaging) in our non-object oriented system.
- They never seem to capture necessary aspects of the system--i.e., the politics or the funding side--only the engineering and technical side.
- Organizational acceptance is at the acronym. Poor training and understanding of the tool produces poor products. Single user acceptance and use does not translate into company acceptance. Lack of awareness of effects and intent of diagrams led to duplicated efforts. More effectiveness would have been available if company was backing the training and user acceptance.
- The model would describe one behavior in the system but was weak in showing how all behaviors worked together to provide the desired solution.
- The model is very effective but establishing a knowledgeable user based that gets the "big picture" and understands the nuances of the model element relations is arduous. So the model is highly effective, but the community doesn't yet understand it.
- - There seems to be no good way to represent a time-based sequence. - The relationship between events, receptions and operations is not intuitive or easily understand. (is this a Rhapsody specific issue?)
- I think the ability to express concepts is there but it requires significant training and experience to appreciate and use it correctly. It could be very easy to fall into a trap of wanting to improve the model for the models sake as you learn more about how to apply SysML. A mature process is really needed at the beginning.
- Visual Activity, IBD and Requirement diagrams provide a greater understanding by stakeholders versus tables.
- There is no obvious formal "executable" language that can be formally verified by proof engine, e.g., FX or TAF.
- Does well in a system level but hard to manage in a subsystem or CSCI level.
- don't use it enough to comment.
- UML and SysML are mostly focus on structure of systems even when they describe behavior like sequence and activity diagrams they do not express a lot of useful information. All in all a very poor notation. Source Code is a much better design notation.

- Would like more quantifiable results out of the model instead of qualitative. Executable activity diagrams would help.
- Our team discovered model-based system design is 49% engineering, 51% art. Many times, we discovered there were multiple ways of representing a design concept (each in its own way syntactically correct), but often not always a team consensus on which "right way".
- Reduce the symbol set, alter views based on paradigm (e.g., security, testing, human interaction, etc.), increase simulation/analysis capabilities, and interface with 3D Studio Max/AutoCAD/etc.
- In our case, our customer started the SysML model and required us to complete it (the customer supplied 1500 requirements). We are required to show satisfaction of all requirements (performance specific requirements) in the functional baseline of the model.
- The ability to setup multiple layer models allowed for highly abstracted views of the system to express concepts. The weakness lies mainly in the ability to inherit interfaces down to some implementation point. Models do need to be translated into other visualizations for presentation to non educated reviewers.
- bdds are more descriptive than class diagrams
- Yes I think they do for the most part.
- The language by itself lacks a flow of the relationships between structural and behavioral entities - this lack needs a method to resolve, perhaps a few best practices would help increase the overall expressiveness by providing context.
- Visual representation of relationships are more concise then written or spoken descriptions.
- I found that I could support both IDEF0 and Object Oriented designs using SysML.
- Along with a good method and tool, the combination of structure and behavior diagrams with formal definitions of model elements is easier to convey the complexity that's hard to show in a ppt.
- Model was rich in expressiveness, although it required several iterations to find the appropriate level of abstraction
- Use Case and Activity diagrams proved highly valuable in capturing the ConOps of the system.
- Hardware intensive efforts and COTS based efforts are not always easy to define and justify
- Yes but the introduction of Ad Hoc networks, and external components within the components of the internal system gets confusing.
- SysML is a static view of a system and has difficulty expressing dynamic nature of some complex systems
- For the sequence diagrams, the models express the message flows.
- System concepts well captured at the Systems level
- The consumer of the model must also be trained in SysML or they will not pick up some of the subtle details of the models. Same for all of the effectiveness measures.
- When completed correctly, the diagrams should capture most of what is needed. A comparison/mapping with DoDAF 1.5 and 2.0 would be very helpful and should also identifiy any viewpoints/models that are not captured and/or to what extent.
- the difficulty lay with representing interfaces/ports/flows in a way that matched our heritage concepts yet complied with SysML

Question 9: Rate how well SysML supports the following language effectiveness measure: Precision - Are the models sufficiently precise to unambiguously specify a system? Please elaborate on your answer.

Results

Where 1= Low Effectiveness 3=Medium Effectiveness and 5=High Effectiveness



Open Ended Responses

- For simple systems, SysML is adequate. For dynamical and or adaptive systems, less so.
- Models can be made sufficiently precise. Whether or not anyone will care to actually use this information is debatable, and will depend upon the skill of the project leader(s). See the comment above.
- Diagrams can be drawn that are not consistent with each other. State charts are not formal. The execution syntax for activity diagrams and state charts is essential to define and get right. Without this it is hard to build a correct model. There needs to be consistency and completeness rules developed for executable models that are allocated to IBD blocks.
- The models are very precise. There is a lot of flexibility built in.
- Doesn't capture all properties of a system, e.g. ilities or those modeled in solid physical model or electrical models.
- To get models precise enough, high effort is required. Consistency is not easily achievable.
- The modeling semantics is not always unambiguous, which is a pity. E.g. questions as, "why did you choose a flow port here?", are not always easy to answer
- Architecture of overall system missing
- For deterministic systems, no problem. For non-deterministic, the more description, the less correct.

- If you arrive at sufficient detail to express parameters then you have sufficient detail. : 1 model not complete: 1 Sometimes too precise for abstract concepts.: 1 The precision is left to the developer of the diagrams. This is a key issue. What level is need? SW, Test, Gov't etc: 1
- There is the model, and there are the techniques people use to explore and understand the model; if the latter are weak, ambiguity reigns, even if somewhere in the model everything is specified. SysML itself clearly provides the means to be very specific, but more attention to ways of exploring models is needed.
- Unambiguous specification can be attained quickly by a team with SysML experience. The key is giving a team the training and opportunity needed to gain then essential experience.
- Also, it does appropriately leave flexibility by not being specific at the higher levels. This gets its information from that, without specifying the data, data rates or digitization (as SimuLink requires--this is an important distinction).
- SAFE GATE
- I see some clients that over-specify the system in the model. I'm not seeing as many clients under-specify the system any more (or miss the critical exception conditions).
- TEAMS deals in a lot of ambiguity -- we did not attempt to precisely specify a system. I suspect, though, that sufficient precision does exist in SysML to do so, if desired.
- I am unsure on whether the specification should become part of the requirements.
- at least in EA - this is simply not there & there doesn't seem to be enough rigor in SysML either - the issue is interfaces
- The problem here is the complexity and difficulty of expressing something completely! There's too much that you can do wrong. The tools "allow" you to do things wrong - and the model validator tells you it's wrong, but not how to fix it. (Why does the tool let me even do that?) I think the biggest shortcoming is the focus of the standard. Too much focus on the diagrams and what can be done, and not enough focus on the information model BEHIND the diagram. What relationships between which objects are acceptable/ semantically correct? Forget diagram semantics - you first have to have the information model (schema) semantics defined.
- when modeling a physical system, the limitation of SysML do not allow for precise modeling of all the required aspects of the physical world, thus leading to ambiguous models.
- Specification of time property requires MARTE. Too many semantic variation point in UML, specially for state machines. Unambiguous specifications require a profile.
- A formal semantics and syntax for behaviour diagram conditions and guards is missing. This does not allow the simulation of a SysML model. Every user/tool vendor has an own syntax defined.
- requirement relations are ambiquous
- Many parts of the language can be interpreted in different ways and there is a need to clarify semantics (this is also true for UML) notably the behavioral constructs and the relationship between behavior and structure
- It is possible to specify very precisely, but it costs readability.
- For our space system applications we needed a domain specific extension in the form of a profile. Initial profile for systems/software co-engineering was defined in R&D study.
- Similar to the above comment. Ambiguity is still prevalent.

- The lack of ability to include instance values EXPLICITLY ON IBDs/parametricts/activities/state diagrams is extra-ordinarily limiting. Precision is not achievable without these and UML instance diagrams are worthless to SEs that don't understand software - besides which the are devoid of communication value as diagrams. SysML is very precise in tracking types and relationships however some exceptions inherit too much from the software world. Inheritance and interfaces are 2 classic examples where UML relies on a compiler to complete the loop as part of source code.
- It is possible to specify the system with a high level of precision. This often decreases the understandability of the model.
- Good results when models and traditional text/graphics and models are combined.
- This largely depends on modeling guidelines and practices not on the language alone.
- SysML has mechanisms for specifying systems precisely.
- we didn't get far enough along to prove it, but I think what we did identified nuances and eliminated ambiguities that had gone unacknowledged in people's minds ... so this is a major benefit of SysML
- Can depend quite a bit on the methodology used and whether or not the models are intended to be executable/solvable.
- The basic issue here is that we can't say: - here's the model: .... - here are questions about this model: .... - here are answers which can specify: a) desired characteristics -- i.e., this is what we want our model to mean in an objectively verifiable way b) actual characteristics -- i.e., this is what we obtain by executing the questions We could specify the questions using, e.g., QVT Operational. This is bleeding edge capability in Eclipse's Model-to-Model project but it's the kind of thing that some tool vendors build using proprietary technology that is, unfortunately, non-interoperable with other tools.
- Judging from the discussions on the RTF, it's not clear that even the experts always understand what is intended by certain language constructs.
- however the inconsistencies between diagram types e.g. actions not explicitly equating to operations - makes correspondence difficult to achieve. alot of manual work is involved synchronizing BDDs, IBDs and act and seq.
- Just think that between State Diagrams and Activity Diagrams there are a lot overlapping. Perhaps more precision than needed but both diagrams present ambiguity. Which one should I apply for what?
- At this point, we are attempting to specify changes to an existing system and that works fairly well. Complete specification of our very large and complex system continues to be a large challenge.
- This was due more to the engineering process.
- Similar to above, the constructs are there to specify the system and its interfaces. Communicating those data outside the model user group is difficult.
- I think that precision of a model is dependent on the skill of the modeler.
- I think the language can be used to communicate unambiguously, but just as you can be ambiguous in English, you can be ambiguous in SysML. I think this again points to needed a well established and documented modeling process at the beginning of a project.
- SysML encourages discipline of standardization, architecture detail and traceability and reduces error.
- The lack of a formal language foundation is a clear limitation on automated formal verification.

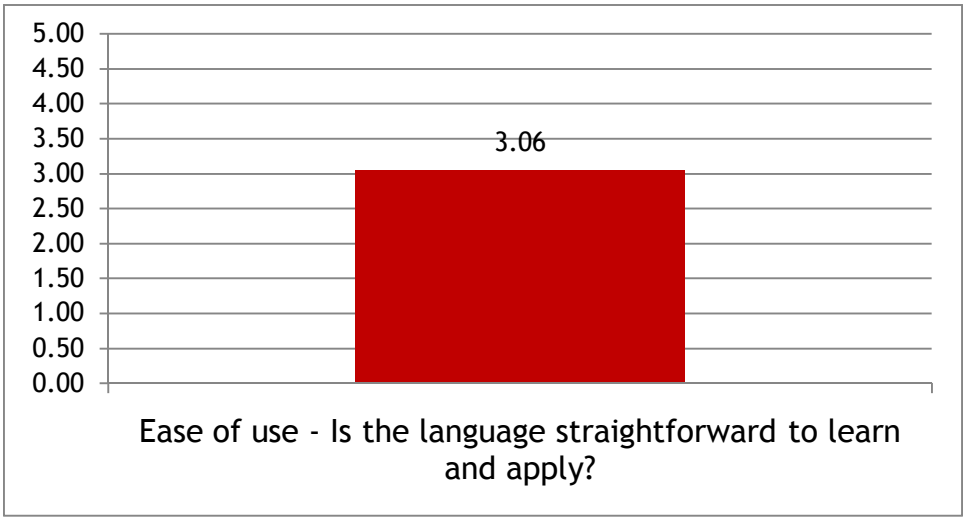
- difficulty in detailing specific configurations.
- Interfaces need more work to be fully precise (need to incorporate a way to record messages)
- don't use it enough to comment.
- UML and SysML do not specify enough detailed information to actual build a meaningful system. Source Code is a much better design notation.
- Easy to get into ambiguity problems with allocations if you are not careful. Again, this may be more of a tool issue than a language issue. I should be able to define allocations within the context of a tool generated spread-sheet instead of placing elements on a diagram and using allocation relationships to update the database.
- This is where the 51% art came into play.
- Keep the set of symbols small and repetitive between tools. I should be communicating my design through elegant simplicity and not because I sent more time learning a tool.
- some modeling parameters we use could not be input easily in SysML (i.e. message size and frequency)
- I think so, but we haven't completed our design to know for sure.
- the model language has sufficient capacity and flexibility to provide precise details, while provide a pictoral view of a design. The balance of the number of objects in a diagram verse providing a precise view of the analysis in more complex systems does require layering and inheritance between model layers which is a little week.
- depends on the implementation, if done correctly, they are precise
- 9. The models should be looked at as a database of inter-related information. There are places where redundancy of specification can still occur where maintaining the model can be an issue.
- still requires people to be knowledgeable in the semantics used by the tool, as well as the language. I suspect the language is more advanced than the Rhapsody tool that we are constrained to use.
- Once a method has been identified, the SysML is very effective in its selection of notation for precision (timing, sequence, states, structure and behavior).
- Captures the overall concepts well, descriptions allow for clarification of ambiguities and exceptions.
- There still room for mis-interpretation. Textual clarification is still needed in many occasions. For example, representing the concept of API (application programming interface) may be hard to grasp by using purely diagrams.
- They could be if done correctly.
- Yes, although it requires significant maturity of use to ensure model constructs are precise
- Since the project was prematurely terminated this measure could not be assessed. However, the early system definition work was well received by peers.
- COTS products have more than one would normally define the system for. Identifying where to limit the model is sometimes a challenge
- Any language has its holes, but I think SysML covers most of them sufficiently. It's a matter of the author clearly developing the diagram and understanding SysML that is the most important.
- Yes based upon what they were defined for.
- Again they represent a static view of a thread, not an entire complex highly dynamic system
- We need to implement more of the models, but we do expect it to specify our system.

- Precision at the Systems level was adequate.
- At the SoS and higher level system levels, should be highly effective. Not sure how well it works for more detailed levels. Although I have not been able to get to this but we need to determine how well the diagrams/models are understood and identify what is needed by the SW engineers and eventually the coders. Need the prototypes to help identify any shortcomings.
- seems like it

Question 10: Rate how well SysML supports the following language effectiveness measure. Ease of use – Is the language straightforward to learn and apply? Please elaborate on your answer.

Results

Where 1= Low Effectiveness 3=Medium Effectiveness and 5=High Effectiveness



Open Ended Responses

- Very easy to use. Supports better refinement during the modeling process as well as informational sufficiency measures within the model product.
- In order to make the language straight forward, we must first make the method (object oriented design) straight forward. There is NO USE in teaching SysML (or UML) if you do not first have people on-board with "Object Oriented Thinking". I have this to be the biggest impediment to advancement, at least where I work. Also, once again, building SysML on the uber-class terminology <<block>> is cumbersome, even to SE's. Those of us who develop data models as part of the system take umbrige at this constraint.
- It is confusing to start with a block and then to create IBD. Why can't I start with an IBD and then build a BDD from it? Systems Engineers think in instance models. From commonality analysis they derive the common blocks/classes. The languge seems backwards.
- Needed the Sandy Friedenthal 2 day course
- Depends on background of staff and the tool.
- Relations (associations, compositions, inheritances, ...) are complicated to be understood and to be used in the right way. Diagrams at first sight can be easily misunderstood.
- For people not acquainted with UML the learning curve is _very_ steep. And even those acquainted with UML can have difficulties (mainly with respect to describing behaviour)
- How do you handle the ArtiSan Sequence Diagram narrative?
- Very easy to create models, MUCH harder to validate the models after they are created. ONE model is almost always insufficient.

- It takes some time to complete understanding of the diagrams.
- At a beginning phase, I prefer SysML to UML2.0 because it formalizes the language to communicate to the SW design engineer.: 1 Basic features very intuitive & easy to learn.: 1 if you are new to UML it is a different way of thinking: 1 Terminology can be intimidating to those unfamiliar with SysML.: 1 The basic elements of the language are very easy. Younger engineers pick it up immediately. Us older guys maybe longer :): 1
- No, because: the technical language of Systems Engineering is a barrier for many stakeholders; the body of tutorial materials needs to go beyond the necessary and excellent detailed narrative found in the book, and extend to concise how-to's and overviews that reference in-depth explanation.
- Significant learning curve, particularly for those inexperienced with UML. Application that utilizes the underlying interoperability of the components (example flowing the data typing), versus using it as a graphics template tool takes a fairly hefty learning curve
- FOR MAKE SAFE
- Learning SysML without basing it on a workflow is like trying to learn German by reading a dictionary. Far too many of the books and courses teach only the language constructs. This is OUR bad. We, as the SysML community, need to fix this problem before users give up using SysML.
- Learning resources are scarce. UML books and tutorials implied that much of the meaning from UML glyphs was dependent on the user. That flexibility made it difficult to be precise with the diagrams. I assumed that SysML was much the same way, which makes it difficult to learn or teach. If it was intended to be more rigorous, a few resources describing that rigor would be helpful.
- Long learning time, especially without previous UML skill.
- For younger engineers who have been exposed to object-oriented methodologies, it is very easy to pick up. For those engineers that have used IDEF before or have used drawings during their career it is a bit harder for them to learn and get used to SysML. They think the use of SysML should be more like a drawing tool and tend to try to go back to IDEF constructs.
- too many nuances
- Not easy to use. Too many software-like expressions. Sequence Diagram is a notorious software-centric expression. As a system engineer, I take offense at anything that I have to define by writing code structures (I'm not building code - I'm trying to express ideas that my customer can understand - most of whom can't understand code!) So, anyplace that forces me to use an underscore "_" in an name, of combine ideas with a parentheses [sendmsg(mymsg)] is not a good expression for a Systems Tool.
- significant understanding of the abstract syntax of not only SysML but UML is required, and this is imposes quite a steep initial learning curve.
- SysML comes from the world of software programming, and many concepts are foreign to system engineers.
- Not so hard to learn from my point of view, assuming that the OO foundationof the language are integrated. Can be difficult for people focused on functional decomposition approaches. Such approaches are supported by SysML but the language can be correctly used for them only if the OO aspects of SysML are well mastered and taken into account, otherwise the model will not be semantically consistent
- After a (short) training.
- language is complex as based on UML and required good UML understanding.

- Non-trivial learning curve is daunting to some (depends on whether prior exposure to UML or not)
- Although from the start it was identified that UML is too complex for systems engineers and simplicity was one of the main goals of the original submission, still SysML is perceived as too complicated by practicing systems engineers. We need to find the right balance between ease of use and expressiveness and precision, perhaps try and find ways to allow users to choose the right subset they need and perhaps give more guidance on the usage of SysML
- Some concepts like activities, use cases, block definitions are easy to use. Parametrics, internal block definitions are very difficult. Extremely difficult is the UML metamodel which is necessary to understand for the definition of stereotypes.
- It is not too bad for people with experience in UML, but there is a very steep learning curve for systems engineers without previous UML experience. It will require dedicated training programs and good pilot project examples to achieve initial adoption. The benefits of using SysML are not obvious to many systems engineers.
- Definitely not. The complexity of UML and the heavy software oriented modeling concepts plus the systems eng concepts introduced with SysML make it a difficult language - particularly for those with no software background. It lacks a strong foundation in SE. SysML needs a clean simple syntax at its core to get SEs hooked on the concept.
- Some concepts like use cases, block definitions and requirements are easy to learn and apply. Parametrics, internal block definitions are more difficult. Very hard to learn for most users is the concept of the metamodel.
- At least it is easier to learn and apply as compared to UML.
- steep learning curve is unavoidable.
- An easy-to-use mechanism for customizing SysML would be highly appreciated so that users can use it by only learning what they need to use.
- more straightforward than UML because of the simplification and extension and it WOULD be pretty straightforward if documented in plain English
- Novices find the language difficult to learn.
- The language is easy to use if one doesn't care much about producing good models. Producing good models requires more careful attention to how the model is organized. This is where the language is short on actually being a "Systems Modeling Language". It would be a lot easier if there were a kind of top-level view of systems modeling an explicit support for specifying the environment of the system in some viewpoint. Right now, a viewpoint involves a lot of opaque strings that don't really help a novice user when they don't really know what to put in such string fields.
- Learning any formal language is not going to be easy for some people. The UML heritage is not necessarily an advantage from the point of view of learning.
- Learning SysML is not a simple matter - especially if you intend to apply it. SysML is pretty easy to learn if you are working in a drawing environment. however, using SysML in anything more rigorous then this requires alot more learning. developing an integrated model of a system in SysML will also bring you to the limitations of the language in its current form. these limitations are not expressive but rather how the various information in SysML integrates between representations.

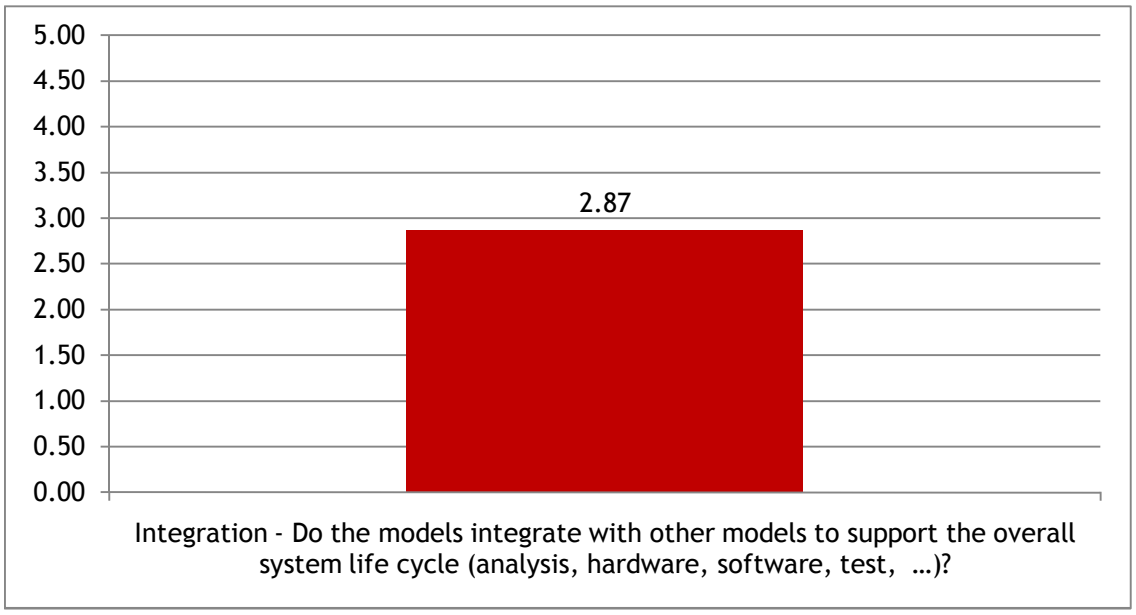
- Activity, bdd, ibd, requirements, and use cases are fairly straight forward at the basic level. Parametric are a not as intuitive initially. Sequence diagrams are often improperly used as data flows. There are often a lot of questions on how to use flow ports, and confusion on the use of standard ports.
- Steep learning curve in general, and subtleties with some of the semantics can cause confusion or mis-use
- The language enables the telling of a great story, however is not very easy to become fluent with the nuances: pins vs. control objects, etc.
- The Rhapsody tool is easy to learn and apply. SysML in total is a large stretch for engineers experienced in legacy design representations and is taking quite a long time.
- Block diagrams for mega-level code takes a good deal s/w development experience to model correctly.
- Have found that in training others to use the language, guidebooks and examples from outside sources are better than the OMG library of language definition documents. Would be great if organization could support a tutorial based booklet or guide (similar to accompanying IEEE Guide to an IEEE Standard)
- There seems to be difficulty in understanding what goals modeling is supposed to achieve. It seems to be more of a 'check the box' activity.
- Easy to learn, difficult to apply well. All the standards aren't yet standardized. Much like with a new programming language there are 10 ways to solve any particular problem and two separate individuals are unlikely to arrive at the same solution.
- Easy to pick up semantics in order to understand a diagram, but application of syntax is harder. Much like learning any language.
- Not really, learning it is not so hard but applying it is very difficult and then generating specifications and interface design documents from it requires writing a lot of visual basic Windows software (not something systems engineers are familiar with).
- The language is fairly easy. The tool implementation tends to caaue most of the learning problems.
- Yes, but like most languages you need to apply it after training before knowledge is lost.
- It's not clear that the underlying data model is sufficiently normalized.
- still some non-obvious areas with ports, types, instances
- don't use it enough to comment.
- Surprisingly for a notation that has so little useful things to say it has hundreds of pages of specification as complicated as any programming language only a lot less expressive and a lot less useful.
- Steep learning curve.
- Based on the number of ill-formed diagrams I have seen unfortunately the language is not easy to learn.
- Without the help of a tool (Rhapsody in our case) - knowing the relationships between elements would be difficult to remember.
- a good methods guidance document is required for any usage and proficiency can be achieved under 6 months of relatively constant use. SysML was used in an international project where party (English and Japanese) spoken language were drastically different. Within 6 months both parties achieved relatively affective communication through SysML.

- any new modeling system needs time and effort to learn and apply
- Some parts are and some are not. Value types, flow ports, parametrics
- Again, I believe the language would be easier to learn given a set of best practice methods that introduce the whys before the hows across the notation.
- In the abstract the language is readily understood but one must use it to truly understand it.
- There is a big learning curve to learn the language to
- It's hard to change from the current practice in my organization where SE practices are not uniformly understood or applied. SE in title only.
- This was a significant challenge, particularly with users who had no previous modeling experience
- The subtleties of the language have learning curve.
- It's amazingly easy to use once you start recognizing the patterns.
- Rather steep learning curve.
- Initially the language seems intuitive but using it effectively is not straightforward or easy. A good methodology and experience is needed to do well.
- The language is complex and hard to learn. But this is to be expected the first time through it. More importantly is to have the SysML Tools implement the diagrams accurately and easily while also not allowing things that are not allowed for SysML. Rhapsody has problems in both of these areas, probably since it is based on UML.
- once we learned UML 2, SysML got easier. It's baffling if you start with SysML. Difficulties with the way different tools implement SysML features
- Hard to find best practices and good examples

Question 11: Rate how well SysML supports the following language effectiveness measure. Integration - Do the models integrate with other models to support the overall system life cycle (analysis, hardware, software, test, …)? Please elaborate on your answer.

Results

Where 1= Low Effectiveness 3=Medium Effectiveness and 5=High Effectiveness



Open Ended Responses

- The MOF or ECore foundations and XMI make this language easy to integrate.
- We have not gotten far enough to make a judgement. I have only done this on simple "seminar problems".
- I always end up with diagrams and sets of blocks that are not consistent with the rest of the model. It would be nice if the language hand formalisms to maintain the consistency between different diagrams. I would like to have a library concept to build models from. Similar to packages but domain specific. Starting with a generic block and applying steriotypes and tags is confusing and full of problems.
- Appears to be very useful with other models. More time needed to investigate.
- Vender dependent
- they do not really integrate in the overall process. Even using similar elements (blocks, classes, parts, instances,..), System Engineering and Software Engineering use them with different meanings (to express different things). It is difficult to find a seamless handover.
- This is one of the pain points of SysML. E.g. it would be nice to define some "entry points" where other models (e.g. simulink models) can be used to described behaviour that is not easily expressed in SysML (or where SysML tools lack simulation capabilities fi)
- We had to come up with our own scheme to link models at level 1, level 2, etc. We still haven't figured out how to create proxies at different levels since double inheritance is not supported.

- There is no clear path to integrate them with other models.
- Yes but requires discipline
- I have identified some possible inconsistencies. For numerical analysis, when LU/SVD/Cholesky work, it's OK.
- SysML adds what UML lacked to complete systems engineering needs.
- Integrate with software models seamlessly. Good luck with other disciplines understanding and utilizing SysML diagrams.: 1 Integration less an issue of language than modeling style.: 1 model not yet finished. recomend improving upon XMI: 1 Our program is just entering this phase..... need more experience to answer.: 1 System and SW engrs should be encourage to participate from the start of a project with SysML.: 1
- Many SDLC models these days involve iteration and feedback loops. SysML can be used to model these, I think, but exactly how has not been explored much, as far as I know--and it should.
- Not at this stage.
- CYCLE WEB FORM
- Very little work has been done to explain the hand off between Systems Engineering and Software Engineering. I see Systems models ignored by both software folks and testing folks.
- Needed improvment with link to Matlab/Stateflow or similar real-time simulation environment.
- Currently we are integrating our models with requirements through a synchronization but are looking at better understanding how to integarte into the simulation portion of SE and how to better use the SysML models for downstream integration.
- not enough experience in this to answer
- Not sure if you mean how models integrate within a tool or between tools. Within a tool, you can create a pretty comprehensive model, with some exceptions that I've elaborated elsewhere. Between tools - no chance. You can't export/ import between tools and get the same results/ model from one tool to another. Obviously, XML is not the silver bullet, since it can be easily re-interpreted differently between tools. Back to my information model comment - must have the standard express this precisely, and then build an ICD with XML semantics on how to map that model into XML, so there is no ambiguity between tools.
- to accomplish integration, one needs a mapping between SysML and the other models, but also an integration between tools for working with these models. This is still cumbersome at the moment, but will improve in the near future as better tool integration becomes part of SysML software.
- Experimentations are on going on those aspects but it seems promising. The usage of opaque expression allows virtually almost any kind of integration the tricky point the consistency of the whole
- Many concepts of SysML are similar to functional model-based development (e.g. Simulink), FMEA system analysis, testing and requirement management and engineering. This makes it easy to use SysML as the integrating element in the complete System Engineering process.
- integration methods are not described and are unknown. UML4SysML subset is artificial barrier for integrations also. Alien models are not allowed in SysML.

- Only moderate attempts made to date to integrate with standard documents (Ms-Word) both on input (import of requirements) and output (generation of ICDs, design docs, various requirement matrices). These attempts so far have been successful.
- As a language SysML offers good integration with software, analysis and testing models. We don't know about hardware models.
- Mostly a tool problem which badly support integration of other models.
- Currently very low integration. Still needs to be developed.
- the diagram-oriented way that the underlying information model/data is constructed is too disjointed. For example parameters defined in activities are different than value properties, which maybe a reasonable for some cases, but when they are one in the same for the model being built – they cannot be combined. Semantic web technologies could help here but even a strong consistent systems engineering meta-model would greatly benefit.
- It is more a tool than a SysML problem.
- Our usage of modeling tools is tailored such that there is a focus on the documentation deliverable. Documentation content is conveyed better when in SysML format as compared to the textual representations in use up to now.
- The only possibilities offered by the language are comments or free text fields with no enforced semantics. We are a bit surprised by the question... It is more a tool issue than a language one.
- Yes, they do, but you need to have a mapping from SysML elements to elements used in particular domains.
- Requirement to SysML tool interface does not intuitively handle the "proper" relationships between requirements and other model elements
- Integration with analysis models was challenging but worthwhile.
- We have done this in prototypes (and in the related spin-off company InterCAX for math solving with parametrics), but more infrastructure is needed to make it easier to do on a broad basis.
- Integration with analysis tools is the most problematic issue with SysML tools. With very poor support for analysis-related OMG specifications (OCL, QVT), one has to rely instead on vendor-specific analysis techniques or make-do with template-driven report generators for example.
- Don't know.
- i assume "the models" means the SysML models". in principle I believe SysML has the capacity to integrate well with other models. certainly SysML surpasses traditional systems engineering representations (documents etc) but there is still much that will be proven out in projects yet to come and tools yet to come.
- tool vendors still need to implement XMI properly. need to develop model interoperabiity methods with other models.
- We haven't used all models to be able to give a fair opinion. However, the combination for example fo a high-level block diagram showing separate activity within the blocks and then supporting activity diagrams for each block is huge.
- We have been using SysML exclusively for "front end" system-level design specification. We see huge hurdles to integrating this with other models.
- dealt exclusively with s/w development, no hardware integration whatsoever.

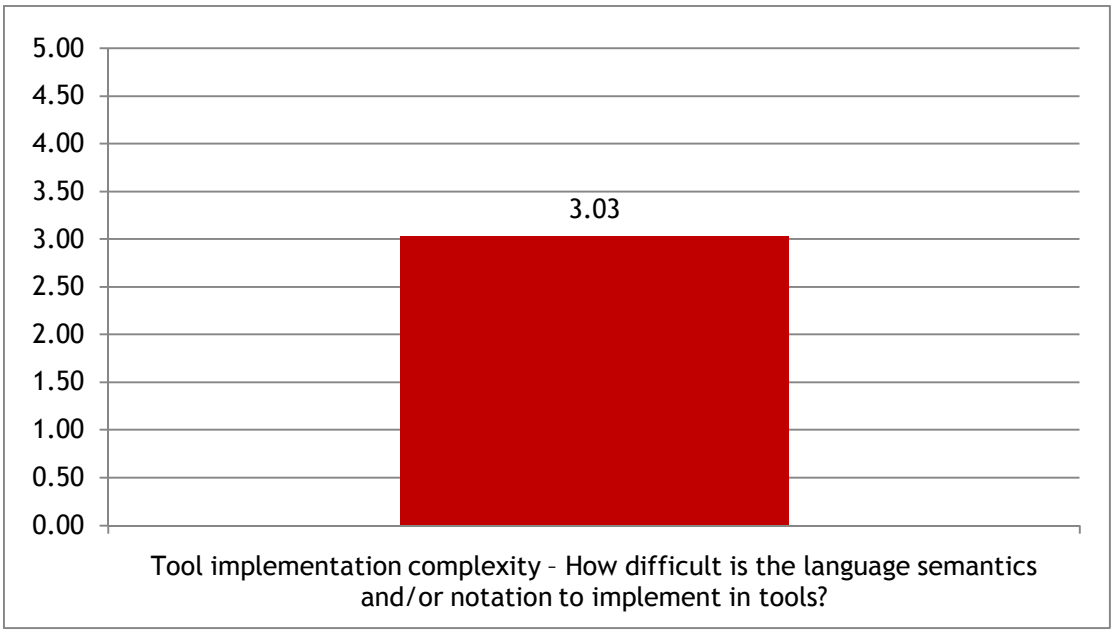
- In textbook practice- this is likely the case. As the models are used by management to suit varying whims, the definition at the top level is not present and lower levels are not used for definition but for documentation of designs already laid into requirements. Requirements and architectures follow design in practice at my organization so have not been able to verify effectiveness of the integration application of the models.
- There were multiple models on the program, but they were not integrated.
- I think in theory the answer is yes. We are working to develop an effective integration. But and have a number of resistances to MBSE both internally and from our customer base.
- This seems to be a tool-dependent question. We use Rhapsody. There seems to be adequate integration with other tools (Simulink, DOORs, Synergy) for our needs.
- Have not had the opportunity to do much in this area, I would like to see better integration with microsoft office products though (word, excel, powerpoint) as well as with tools.
- having difficulties how to incorporate other embeded system models developed by tools such as Simulink, MatrixX ....
- We have little experience with model integration with SysML at this point, but believe that it will integrate with software and test.
- inconsistencies with UML makes round-tripping with S/W more difficult; tools are insufficient with XMI. Where is AP233?
- They really only integrate through hyperlinks to files... There is still no way to actually pull in information from other tools or directly link values from tool to tool.
- Don't have a good way to link requirements and test procedures, hard to manage in SysML compare to other tools such as DOORs.
- don't use it enough to comment.
- So much time was spent on developing fluffy diagrams that a lot of the hard engineering work was avoided because we ran out of time.
- Difficult to transition from SysML to final product -- transition manual.
- Reasonably good integration with requirements analysis. No support for system availability (based on hardware and software reliability) modeling (quantitative projections).
- It is difficult to develop a layered (general to specific) presentation along with automatically alter the view of the design for different paradigms (e.g., security, testing, requirement change impact, etc.)
- Mainly a tools issue, being able to integrate data across tool sets. We were able to use SysML to represent a design which could be remodeled in a simulation tool, with the model diagram providing all the data.
- testing state charts are valuable to verify code operation
- It is imperative to have an architectural model coupled to simulations, spreadsheets, test and design models. This is one of the true values of the model.
- too early to tell. It should, but we are not experienced enough to evaluate.
- I have not crossed the SysML / UML bridge as yet so I cannot rate this integration, however, from a requirements traceability perspective, the allocate relationship supports most integration goals (SW to HW, Logical to HW, Requirements to Use Cases and Activities).
- In theory they do. However, the tools currently available for creating the models do not interoperate well.

- It seems to be a lot of additional work to pull in supporting documentation (e.g. Excel, Matlab, C++/Java, etc.) Using the "tool" becomes an occupation unto itself.
- Don't have experience with it.
- They could if done correctly. For example, I am able to create DoDAF models and then capture a specific diagram in SysML to begin the process of design and analysis there. However, the customer needs to be taught how to move between the two languages. My customers did well on two different projects. Same for going from SysML to UML.
- none of the projects I worked on were to that step of development; but where we did consider integration (based on parametric diagram) no tool integration was available at the time either. How the parameters relate between system specification and analysis was effective.
- We achieved only a rudimentary integration of multiple models. The language supports integration, but we did not fully conquer the required learning curve.
- Since the project was prematurely terminated this measure could not be assessed.
- Issues with interfacing to other systems and to manual steps is always a challenge
- Yes and no. This is where the use of structures is becoming an issue. The use of DoDAF is great for the 50,000 foot view. But at the 20,000 foot level it can be made to work. AT the lower levels it gets cloudy. SysML works at 30,000 feet down. Any upper level I have found that the natural progression is more complex because it is less system and more behavior at the conceptual level.
- Only experience was integration with DOORS
- We are working on it but there is not just one solution.
- Need to have a better method beyond allocation to map Flow Ports to Standard ports for System to Software/Hardware Interface Definition. Also, Can not automatically specify flows as Primitive Operation Arguments - no drop down list Use of Flows. The aforementioned are Rhapsody issues.
- This is heavily based on the tools that are used. It is very difficult to determine an answer to this question just by analyzing the SysML spec. If the tools implement SysML exactly, then a developer could figure this out. Believe that the tool developers could probably answer this question better.
- We explored connecting tools - Rhapsody -> DOORS, ClearCase... worked well enough within a product family. We looked at accessing the SysML object model via vendor-supplied API, with intention to extract data which could be fed to performance modeling tools. The project was terminated before we got that far.

Question 12: Rate how well SysML supports the following language effectiveness measure. Tool implementation complexity - How difficult is the language semantics and/or notation to implement in tools? Please elaborate on your answer.

Results

Where 1= Low Effectiveness 3=Medium Effectiveness and 5=High Effectiveness



Open Ended Responses

- Most modelers are FAR too timid in using the language and "blame the tools". The complexity often exists in the mind of the modeler.
- I do not know if this can be made simpler. A picture can be simplified if reduced to one pixel, but then it contains no information. Simple is over-rated.
- Some are better than others
- Not done
- Inconsistent XMI is a pain and makes translation into Chi and other sim languages consistently difficult.
- I have not used SysML in a tool yet.
- NA... I don't build the tool: 1 Rhapsody implemenation is not consistent with SysML definition. Feature and terminology disconnects.: 1
- We are not trying to implement SysML in a tool.
- TOOLSCORBA
- I see some Systems Engineers complaining about the tools complexity, but in reality, they are complaining because what they are trying to do is either (a)not supported by the tool or (b) not correct SysML. Again, the solution to this problem is to give a workflow that is supported by the tool.

- I had problems getting EA to draw what I believed were valid SysML ibd diagrams. I couldn't get the ports to draw on the frame, so I had to invent my own notation to subsitute.
- Huge differences between different tools. I evaluated Topcased, MagiDraw, Enterprise Architect, Artisan Studio Uno.
- comments above about EA being too flexible -
- Tools vary all over the map on how the enforce or do not enforce certain rules. Some enforce specific rules for diagrams (which I find most frustrating - why can't I add a Use Case item to an activity diagram, if it helps me express an idea?) and other tools allow anything to happen, but you're not sure if "it's the right thing" to happen. on top of that - none of the tools that we've tried have any Help information regarding SysML!! "What does this mean?" Nothing - and very little available on the web for support to answer questions.
- takes some getting used to, but once familiar with the tool is quite easy and fast. property specific types are VERY cumbersome and need to be revisited.
- The SysML profile is easy to apply. We have defined our own SysML profile. This allows us to enhance the standard by own additions for our application domain (automotive).
- some arguable UML notation changes are unnecessary overhead for tool vendors.
- Being a UML profile it is not hard for UML tools to offer support to specify SysML models however it is quite hard to provide tools that fully allow model execution of the models.
- Metamodel is complex; Main problem is a profiling of the tool and the user interface.
- SysML being UML2/MOF based is quite difficult to implement. Much more success experienced with a lighter Eclipse/Ecore based MOF in prototype implementations for Virtual Spacecraft Design R&D activity.
- easy to teach and learn, but at times their is a blur between UML and SysML languages
- The metamodel of SysML is complex. The main problem for tool vendors is to implement convenience functions and to allow a profiling of the tool.
- We see too much software engineering heritage in our SysML tools.
- We do not build SysML/UML tools.
- We haven't done this, but probably it will be easy.
- the tool we used, EA, was fairly pleasant to use, but there were a lot of fussy detail issues that ultimately interfered with our ability to "sell" SysML in the organization
- Tools don't seem to fully compliant yet. Experienced a lot of frustration trying to figure out how to get tools to do what the language requires.
- The SysML specification itself is incomplete; for example, it lacks most OCL constraints. The specification doesn't provide any verification criteria that would perhaps give vendors some incentive to use for value-added capability for end-users.
- although I am not a SysML tool implementer - I would notice that as a language there do not appear to be any fundamental open source tooling for SysML. in software this has been useful for universities and more theoretical application in software languages. this could be overcome with a stronger and tighter information model behind SysML. this would allow open source tools to be developed more easily and models to be more portable.
- We wouldn't be able to gice an input at this time; we haven't used most of the features of the language yet.
- Almost takes a tool training course to understand all the features. Luckily, I had a lead who knew showed me the ropes

- Non project related - In studying the UML spec, there seems to be inconsistencies in the language and it seems to be overly complex.
- Some good some not so good. We've worked exclusively with Rhapsody which does not yet fully support the SysML standard. Flow ports and flow specs are particularly tricky.
- AS mentioned before, activities and actions are hard to use effectively. Activity diagrams cannot be dragged from one place to another (this is true in 7.3, untried in 7.4, 7.5.) Action block decomposition-- it would be nice to be able to be able to use an action block to represent another activity diagram, with the pins corresponding to object nodes. It would also be nice to be able to put swim lanes within this sub-activity diagram, and link multiple action-blocks to this sub activity diagram. Also, states as model elements, as mentioned elsewhere.
- Not sure this is a language limitation, maybe a current tool limitation. It seems like there is always another tool.
- It is time consuming to thoroughly define ports, this needs to be made easier with tools such as Artisan.
- System Architect 11.2 does not have a well developed full implementation.
- i'm sure inconsistencies with UML make implementation more difficult for UML tool vendors; Need more table display and editing capabilities; automatic and standard view creation
- I do not implement tools, and therefore cannot respond to this. But based on the tool capabilities I've seen, it must be fairly difficult!
- don't use it enough to comment.
- Tool dependent, for example: Rhapsody - difficult Enterprise Architect and MagicDraw - less difficult
- We used Rhapsody and the tool it was nicked named Crapsody. Crapsody is a bad idea poorly implemented.
- Rhapsody 7.3 has all sorts of problems with SysML. Overall the biggest problem is that it performs no error checking on model constructs. - Cannot apply activity parameters to activity diagrams - When a block operation is placed on an activity diagram as an action no checking is done to ensure the number of pins and parameters between the operation and the action is consistent. - If I have a flow spec that is typed that I am using as an interface between two blocks I can place flows between the flow specs that do not correspond to any items specified in the flow spec and no checking is done. - There are many, many others
- Ideally, the tool would help construct/analyze use cases and then assist in identifying (or even auto-complete) portions of the design based on previous models or libraries of models.
- runs well and is used in our ICE development environment
- we don't make tools.
- I have used Rhapsody with implementing SysML so far and am generally pleased. I am concerned with cross tool reuse, however, but have not needed to explore that to date.
- Rhapsody is generally effective although flows are clumsy in a CM environment, inherited operations/interfaces are problematic on SeqDiag.
- This question is interpreted to mean, "How easy was it to develop work-arounds from tool limitations on the language?" Not too bad. Mostly a matter of documenting the work-arounds and creating some stereotypes to fill the gap.
- Training with practice
- We used Tau/UML which has the primary semantics already implemented.

- There are numerous tools out there but each one has their funny little quirks for SysML. Price, hard to use, etc.
- We are using Rhapsody and have had a few problems that needed to be worked with the vendor.
- Steep learning curve.
- We started using the tools early and struggled with non-compliant implementations. There is definitely a learning curve for both the lang and the tool of choice as well as establishing best practices for both.
- Need to have a better method beyond allocation to map Flow Ports to Standard ports for System to Software/Hardware Interface Definition.
- Tool vendor question. I'm not one.
- We found the learning curve steep for Rhapsody, Tau, and Rational Software Architect. not easy tools to learn, many quirks and "features"

Question 13: Are there other effectiveness measures you would use to assess the effectiveness of SysML? Please list and explain.

Open Ended Responses

- Is any other foundation than UML (tools) being considered?
- The effectiveness of SysML in modeling is usually not very constrained by the language or the tools, but in the ability of the engineer to express concepts and ideas using the language and the tools.
- Implementation (tool vendor) dependent item: Models must be visible to those who will use them, like mechanical drawings. Right now, the SE is the bottleneck. Visibility tools (or other presentation tools) would be nice. Perhaps some automatic way to create presentations which will be accessible to all?
- Develop a collaborative web site (under SysML Users) for posting of success stories.
- Completeness of a system specification.
- Number of integration problems that occur in a multi-disciplinary design
- Artisan Sequence Diagram is really like detailed Use Case. Rhapsody Sequence Diagram is reall whole different animal
- Yep. Methods for relating the models to data and error - so-called Bayesian inference - otherwise models could be inconsistent or meaningless.
- Not applicable.
- Easy acceptance by SW engineering has improved communication. : 1 Number of projects adopting, number of contracts requiring...: 1 SysML training, lots of good examples: 1
- WEBSITE
- At the larger Military/Aeronautic/Astronautic companies, the System Engineers specify the interfaces between subsystems down to the bit/byte level. We don't support that in SysML because we say that the Systems Model is the requirements, not the design. We need a way to address how Systems Engineers create ICDs.
- Traceability is a key feature to facilitate requirement management. When I modify an element in a diagram, it should be easy to evaluate the impact in related elements.
- It would be nice to have some way of reusing pattern designs of systems
- Ability to abstract the non-essential details from essential views. Ability to generate a complete model with multiple perspectives (almost there, but ...) (most examples that I've seen are incomplete models - a few different perspectives, but little unification between perspectives - nice drawings, but how are things related to one another behind the drawings.)
- duplication of other modeling work done by software engineering and older SE techniques. SysML often not seen worth also doing.
- No other measures for now.
- It is very important is to have sufficient training materials and worked example models available for the relevant industry sector. This is currently not yet the case, but is essential for roll-out to real projects.
- Modeling extent - the breadth of topics that an be modeled by SysML.
- Up to now tool vendors have only implemented parts of the language in their tools. It would be nice if a tool implementation effectiveness measure could be created and maintained.
- Mapping of graphical representation to textual for document generation.

- Flexibility or expandability for defining domain-specific models.
- A reproducible verification test suite of SysML models exercising well-formed models and ill-formed models as well as various analysis & results.
- Better harmonization of concepts with the field of knowledge representation, better integration with semantic tools and system that support inferencing, reasoning, proofs, etc.
- I would suggest correspondence. since information can be common between diagram types. productivity: SysML could cut time and money out of getting a good spec and maintaining it. measuring how productive people would guide that. for productivity I would have suggested that static package structure seems convenient at first but refactoring becomes confusing. ultimately I discovered was that I would like to have more than one organization of packages.
- I would try explaining it to novice engineers and see if they get it. If they do then it's a good way to validate the language.
- In terms of a metric? I suppose the level of understanding reached by the team could translate into the n umber of requirements traced into and out of the model. Also, software test cases built on the usage scenarios rather than traditional regression testing could also be a measure- the scenarios developed in the SysML model should be continued in active system use- not archived when completed as a system artifact.
- Are model artifacts linked to project risk assessment? How often are model artifacts used to evaluate project issues? When requirements, architecture, design, implementation or tests are changed is the model used to identify the impact through out the system?
- Does SysML improve requirement rational and flow down from Systems Engineering to Software Design and Engineering? Does this reduce programmatic risk?
- Can't think of any.
- Reduction in system design errors, suitability as a metric in sizing systems engineering projects, support for collaboration among geographically distributed sites.
- Yes, measures on time to develop and review diagrams.
- Ease of formal verification (LH/proof).
- repeat this survey in a few years and compare results
- Don't use it enough to comment.
- Ability to transition SysML artifacts through the phases of a program without manual re-work/translation into other tools.
- Connectivity to other tools (e.g., AutoCAD, 3D Studio Max, Adobe, etc.) for modeling and simulation.
- 1) Ability to export and share data across multiple tool sets 2) ability to reuse objects across multiple diagrams 3) object inheritance (interfaces as an example) through multiple layers of abstract (decomposition)
- The model helps to bridge the gap between software developers and system engineers
- One problem in the area (I have encountered this is discussions of Enterprise Architecture at the high levels in mostly academic environments. But the Transform from a purely business (which includes changing structures of a company or agency) and then relating it to all systems (not just IT) and then interrelating IT to the business and the layers of business becomes very confusing.
- The flexibility in depicting concepts caused lack of uniformity among the physically distributed team.

Question 14: What additional capabilities or features are desired of the language? Please indicate the importance of the additional capability or feature by placing a 1 (low importance) to 5 (high importance) after each new capability.

Open Ended Responses

- The temporal (esp spatio-temporal) domain consisting of time and frequency needs to be improved. We have a solution that works for us, however it may be unnecessary for most simple systems engineering.
- Time based activity diagrams. 3 The concept of a domain specific library of blocks that are used to build a systems model. 4 Formal non-ambiguous execution semantics for Activities and state charts. 4 Hierarchy 3 Model exchange standards for both the model and the diagrams. 5
- In the Block Definition and Requirements, would like to see reliability as an inherent property. Would also like to see Testing Method as an inherent property
- Prompting the users to provide tips on next steps. Ability to refactor models easily, even offer refactor options (cohesion and coupling, complexity indicators, etc.)
- Nested Interfaces: It would be highly desirable to have the possibility to define nested interfaces associated to standard ports. This would allow to define an overall interface on a port, containing subgroups of interfaces. This way all operations of those contained interfaces would be provided or required at that port. without having to list them (or to list all the operations).
- 5/ More Explicit semantics (e.g. enabling simulation) 5/ Better tool support (I know, this ain't a language feature :-) 5/ More elaborated examples
- UML Timing and Communication diagrams: 4 Block instances: 5
- Paths for integration with other models and other stages of system development. For example, how to go from SysML system of systems model to UML software system model!
- I like Artisan Sequence Diagram. Appeals to System Engineers. Rapsody Sequence Diagram appeals to software. Maybe need both.
- Explicit mapping to a formal mathematical graph construct.
- 5) Use of Allocation Tables to implement SRVM... I don't see this yet. Then export to DOORs where required.: 1 Action semantic/executiability, dynamic linkage to analysis tools, templates & profiles for bridging to Sw: 1 agree with SE^2 team MBSE challenge http://mbse.sysmod.de/: 1 Timelines are critical. Sequence diagrams help, but don't address all needs.: 1
- Italy Ingles Spanish Portuguese BR Ingles American
- Improve interfaces definition. Direction of relationships are quite difficult to keep in mind. Clarify the hierarchy structures, especially when it involves more than on diagram type.
- Requirements managements 5 Semantics to external systems interoperability 4 More learning free contents to a rapid adoption from systems engineering community 5
- 1. Would like to see the integration of Test Concept throughout the language, similar to the requirements diagram, the test diagram would relate the integrated test requirements throughout the model.

- Tabular views (5) Interface Control Document support (5) - ability to build the ICD from the information flows - including data item, software data stream, electrical data link, and mechanical linkage (basically, support the OSI layered communication model, with each layer represented by different views/objects. Cohevise Sequence and Activity Diagrams (bind them) (4) Simplify ...simplify... simplify ... and get rid of confusing terminology! There are several concepts trying to be represented by different object types that could be easier to represent as one object with different attributes/ properties (activity vs. action - why need distinction? merge them, and make an activity with no further decomposition an atomic action!) Why are we using the term "property" to represent a "part" - when tools use "property" to mean "attribute" of an item, too. MAJOR CONFUSION to teach someone that to create an instance of a block you create a "property" of that object on the IBD. Huh??
- Need to move SW and SE together into modeling language.
- 5 -- requirements need properties not just text 4 -- to support reusable libraries, we need property specific types that are easier to use; once they are easy to use they could also be often used instead of instances
- 5 - support of physical flows 5 - support of data broadcast 3 - support of periodic messages
- Explicit support for synchronous domain (eg. SCADE like) => 5 Allocation without dependency (for alignment with MARTE) => 4 Action language (from UML ?) => 4 Formal concrete syntax for SysML => 5
- Currently, the capabilities of SysML concerning requirements are limited. Modern Requirements Management tools offer more functionality. A specification on how to exchange requirements (called Requirements Interchange Format) has been developed in the manufacturing industry and may provide ideas for the future development of requirements related concepts in SysML. A mapping from the Requirements Interchange Format to SysML can be found at: http://www.omg.org/cgi-bin/doc?mantis/2009-08-01
- - Support of Functional Safety Management (FSM) e.g. SIL levels as a tagged value for components and requirements - Special stereotypes for software elements and software data types. We introduce a "Data Value Type" as a combination of Data Type and physical type. - Reference concept and reuse (Avoiding of modelling identical parts on more than one place) - Better support of behaviour modeling and timing. Activities have no possibility to specify and change the duration - Better correlation of elements in different diagram types. E.g. a data type defined in a block can be accessed and changed in a behavioural diagram. - Tool support of complex data and element structures defined using associations. Usage of the "Point"-Operator e.g. House.Room.SizeX = 5
- 5. parametrized requirements (numeric value/requirements holders) 4. connections among block ports and behavior parameters 4. SysML-MARTE integration 3. SysML-AUTOSAR integration 3. add more simulation and execution oriented concepts.
- Not yet experienced enough to provide meaningful input (using for only ~8 months). To date most of the "capabilities" found lacking are more to do with the tool's support (or lack thereof) for specific language constructs than with the language itself. Perhaps a language issue is the lack of a way to differentiate a port on an outer block that is truly representative of a junction at that level of the decomposition vs. a port that is only a logical/abstract port representing the port on the lower (but currently not visible) part.

- Product Lines Engineering Support: Support the design of a family of products in a single model (importance: 4) Support modeling of data flows between blocks and parts (non object oriented, in a traditional block diagram style) (importance: 4) Support modeling of part hierarchy to represent parts in a context of a specific composition (importance: 4) Enhanced support for trade studies and optimizations: Extend SysML to support the notion of alternative designs to perform trade studies (importance: 3) Standard constraint language to specify mathematical constraints (Parametric Diagrams): Have a simple and intuitive language to specify mathematical constraints (importance: 3) Support the concept of layers - architecture, functions and protocols, for example support modeling of network protocols or support layered design approaches (importance: 2)
- 5 - Modeling of multidimensional aspects 4 - Multilayer modeling 5 - Nested ports 4 - Structural ports 2 - Named comments 4 - System context diagram
- - Standardize interactive tabular views (i.e. not just generated tables), Importance 5. - Integrated support for instances of <<blocks>> and <<valueType>>, better instance/value specification language. Importance 4.
- I am wondering if it UML and SysML should be one language.
- 1a) Instance values on activities, IBDs, states 1b) remove ambiguity related to software source code specific modeling 2) define uses of states, activities and parts on all 3 diagrams 3) editable table views for all diagrams and/or a separate diagram type of “table” 4) editable text version of the SysML with a complete mapping to the graphical elements.
- 5 - Nested ports: The abstract syntax already allows nested ports by typing ports with blocks which owns ports. It is necessary to define a clear semantic of nested ports. They help in better re-use of interface definitions. 5 - Integration of Activities into parametrics diagrams: bind parameters/attributes of activities to constraint parameters. 4 - Connectable Flow Properties: When typing a flow-port with a flow specification the individual flow properties cannot be connected in the IBD to flow ports of nested parts. This can easily be solved by making flow properties simply flow ports. This would unify the flow property/port specification and is an easy change in the spec and in the tools. 5 - Physical ports: Ports a special parts of a block located at the boundary to interact with the environment. They could be logical, but also specify physical interaction points like sockets. 5 - Junction ports: Junction ports represent ports at the border of a block which are located in a nested part of the block. With junction ports it is possible to specify and show all interaction points to the environment at the detail level of a block. A specific Icon is needed to avoid cluttering of the diagram with stereotypes. Semantic is needed to define delegation capabilities to nested parts. 5 - Property-Specific Types use fully qualified path to redefine a deeply nested property, without the necessity to redefine the complete hierarchy as specializations. 3 - Requirements with attributes required for seamless integration of requirements by binding parameterized requirements to constraint blocks. 3 - Pin to Port allocation Allocation ObjectFlow to ItemFlow The ObjectFlow (Edge) describes that in the context of an Activity the output of one Action is bound to the input of another action. In the context of a block a item flow describes the flow of an object from one part or port to the connected part or port. The allocation of the ObjectFlow to an ItemFlow defines which ObjectFlow corresponds to which ItemFlow in a given context. Supplier and producer and context need always be defined. Allocation Pin to Port The Pin defines which objects flows in/out of an action from a functional point of view. The Port defines which object flows in/out of a block from a structural point of view. The allocation of Pin to Port defines the mapping of functional to structural view, independent of a context. The supplier and producer need to to be known, e.g. when certain data flows over an Ethernet port but it is irrelevant

who is connected to it. Details see http://mbse.gfse.de/documents/se2_rfi_SysML20.pdf

- Simulation of activity diagrams, prio 4 (would decrease the usage of sequence diagrams - Well defined Modelica semantics, prio 4 - Graphical distinction between physical flows and data/information flow, prio 5 - Support for capturing assumptions/promises on ports and pins 5 (as defined by the SPEEDS project) - Integrated version and configuration management support, prio 5.
- A mechanism to define domain-specific models, 5.
- Maybe, a way to easily organize levels of elaboration or detail that develop from one diagram
- a) Support for a full set of rich collections and their related capabilities (e.g. as in Smalltalk): bags, lists, sets, etc. ordered/unordered, ... [1]. b) Model library management conventions (e.g. library naming, etc.) [1] perhaps using something like a java package naming style (com.acme.xyz). c) Better support for instances, including instance libraries. Instances are key for MBSE during model development time, not just run-time (much more so than for traditional UML / software development) [1].
- 5 - user-configurable models of execution for activities & state machines. fUML is great but fUML is only as good as one has support for specifying the particulars of the execution environment. Without the flexibity to specify what such particulars are, it would be ill-advised to include in the specification an arbitrarily-chosen execution model -- one size doesn't fit all purposes. 5 - QVT support. This would be very useful for integrations between SysML and other popular modeling languages, e.g., Modelica, Simulink, and even proprietary modeling/analysis/simulation languages. 5 - Diagram Definition. We should be able to specify, normatively, the SysML diagrams by a QVT transformation from the SysML abstract syntax to a DiagramDefinition of the SysML diagrammatic building blocks.
- 5) -) correspondence between parametrics and act/seq/state in terms of parameters, states and state variables, time variables and timeline of behavior -) better correspondence between act/seq/state for behaviors and states and exchanges -) better correspondence between act/seq/state and ibds in terms of ports/interfaces/item flows 5) real time representation becoming a first class citizen. 5) capability to deduce ports/interfaces from behavior models. 3) non-static package structure. 3) a SysML ontology 3) a text reprsentation or version 2) a concept of model rendering like auto-cad. where 3d vs wire frame might be icons and color schemes vs detailed SysML.
- 1. timelines for activity diagrams 2. improved execution semantics including inegration of structure with behavior (e.g., item flows, flow ports, paramters, pins in activity diagrams) 3. more effectively modeling of property values (properties values vs time, data sets, required vs actual values, ...) 4. ability to more effectively model variance of design configurations andd product lines 5. Ability to associate image with model elements and enhance graphical representations 6. Built in capability to capture metadata such as a description filed for each model element. This can currently be done with a comment, but is not part of each model element. 7. Built in capability to include external reference such as a URI 8. diagram interchange 9. Standard SysML metamodel that is derived from UML4SysML and Profile, but can stand alone and used for transformations to other models
- We won't be able to give input at this time given our rather little experience with the language.
- Rhapsody is the tool now used for SysML. It is not yet robust or mature enough to accomplish the DoDAF views gracefully. I especially wish that the sequence diagrams could have steps on the left hand side that could be linked to requirements in DOORS.
- Worked with SysML for 18 months so I was focused on how to the the design out the door as oppose to looking for new capabilities. I did this almost 4 years ago as well.

- A prescriptive syntax that links artifacts within the model should be added to the language. This syntax should be required of all tools, but the ability to disable all or part of the syntax should be available. Some suggested rules that could be considered are: Rule: One activity diagram per use case. Purpose: A single AD should show all possible path related to a single behavior. Each path should be seen in context of all the other paths. The activity diagram is the heart of the use case and allows the requirements traceability matrix to be auto generated. Rule: Identify each "Path" through the activity diagram Identified. Purpose: Each path will be linked to both a sequence diagram and a written scenario. The activity diagram is made up of multiple behavior paths. Each behavior path will be related to a sequence diagram and a written description. Rule: A sequence diagram for each path (scenario) in the activity diagram. The sequence diagram name should match the path name. Purpose: A sequence diagram will identify the interfaces associated with a single path of behavior Rule: A written description for the each scenario (linked to each path in the activity diagram and to each sequence diagram. Purpose: The written description will identify the requirements associated to a single path of behavior It should be noted that in this process the activity diagram, and all the sequence diagrams and written description will provide the same information. They simply present the information from a different view point. Rule: Each activity, decision, fork, and synchronization in the activity diagram should have one (and only one) low level functional requirement linked to it. Purpose: Reduce coupling between stakeholders to interface coupling only. This allows the assignment (linkage) of a single stakeholder responsible for specific requirements. Identifies where there are duplicate requirements. Rule: Each low level functional requirement should have one (and only one) activity linked to it. Purpose: This reduce coupling between stakeholders to "message coupling" only. Helps identify where derived requirements are needed. (See example, Table 3, Table 4, and Table 5) Rule: The activities link the requirements to the stakeholder (IPT, CSCI, HW, Sub-contractor) that is associated with the swim lane the activity is in. Purpose: This allows for automatic generation or stakeholder requirements specification. It also allows for the validation of stakeholder responsibility related to each requirement. Rule: Each control/transition that crosses a swim lane in the activity diagram should correspond (and be linked) to an interface in the sequence diagram Purpose: This allows for the linkage of each interface requirement to the functional behavior (requirement linked to the activity) that triggers it and to the functional behavior (requirement linked to the activity) that the interface triggers. This also allows the cross checking between activity and sequence diagrams to check for inconsistencies and the justification and validation of each external and internal interface in the system. Rule: Each statement in the written scenario is linked to an activity in the specific path in the activity diagram. Purpose: This allows for cross checking between the written description and the activity diagram. Rule: A lower level reusable use case can be used as an activity within an activity diagram. Purpose: This allows the definition of common behavior. This rule should not be used to simply the description of behavior as all the behavior should be seen in context with all other behaviors of the system. Rule: External actor swimlanes do not contain any activities with functional requirements linked to them. Purpose: The external actors are not part of the system under development. It is acceptable to have 'will' statement linked to the external actor activities. The 'will' statement should become agreed upon (contractual) behaviors and/or interfaces that the external actor must fulfill.

- The language definition seems not to be a significant barrier today. Tool vendor implementation, experienced modelers and System Architects who understand the value of MBSE and the role SysML plays, as well as meaningful large-scale program implementations that can be pointed to for example are the current bottleneck.
- 1. Action decomposition is a necessity. Although the OMG spec allows activities to be classifiers, this is not enough for a one-to-many decomposition of actions. We have a critical need for actions to be classifiers.
- I believe in order to be successful the products developed in teh tool need to be easily exportable to specifications, interface design documentation, architecture description documents and design review presentations slides. This is a significant failing of SysML tools, perhaps the language needs to have requirements in this area.
- Need the capability to interface IBDs to schematic capture tools.
- Support for automated formal verification - FX or TAF. Export of Z-language or SCR specifications.
- Qualifiers, protocol stms, more consistency with UML (3) SysML-marte integration (4) ability to link sd activiation regions with activities (4) parameterized reqtms; requirement templates (5) better organizing constructs (4) better support for automatic view generation/regeneration (4) support for mis-use cases/threat analysis (3) Timeline requirements/analysis/diagrams; integration with activity/sequence diagrams (perhaps with MARTE) (4) consistent external references to/from model capable of rendering/importing (3) diagram interchange (3) better ability to show several specific configurations (5) ability to specify properties over time by connecting to parameterics (3) support for periodic messages (3) Intentionality modifiers for diagrams,etc. that is to be able to identify consistently whether a diagram is indicating a fact of the world or a requirement to be enforced. For example, on a sequence diagram you can use an ASSERT operator indicating that fragment is the only possible sequence -- but it is not clear whether this is the only sequence that can occur vs the only sequence that the system should allow (5) better (and settled) units/values/dimensions supporting standard and non-standard unit systems. When done, standardized unit libraries can be supplied (4)
- Message definition (5)
- I wish UML and SysML would not be used as much as it is. It seems to be mostly used as an excuse to avoid hard engineering work.
- Ability to transition SysML artifacts through the phases of a program without manual re-work/translation into other tools.
- Make the activity diagrams and state machine executable!!!
- System availability (quantitative) projections.
- Some modeling parameters we use could not be input easily in SysML (i.e. Message Size and Frequency, Message processing time)
- 1) affective method for using Use Cases with Requirements 2) Functional Block definitions and view for IBDs
- Better linkage to DoDAF models 4
- 1. Block Inheritance – A more formal mechanism is needed within the language to define what block properties can be inherited, and to what extent can each inherited property be over-ridden. As an example, the following properties of a block should be inherited and certain attributes of these properties should be capable of being over-ridden that is change their type or value without changing the parent block's associated type or value. 1. Value Properties •

Initial Value • Type - Override with subclass 2. Reference Properties - Override type with subclass 3. Operations – over-ride arguments with a subclass, same signature, and different description 4. Parts – override multiplicity and ordering, override with a subclass, 5. Ports - over-ride type with a subclass, name 6. Block Description – override text where appropriate 7. Any other contained model element, including requirements 2. Requirements Mapping – Dependencies are used throughout the language to relate requirements to other model elements. Derived requirements should belong (containment relationship) to modeling elements that satisfy them and that realize the describe functionality. They should therefore be capable of being inherited and allow properties on them to be over-ridden where appropriate. 3. Crossing Lines – A model icon should be defined that clearly indicates when crossing lines are connected or not. This is very useful on activity diagrams and IBDs. Examples can be found on most electrical/electronic wiring diagrams and schematics. 4. Actions should be capable of being contained within the block like operations and attributes. Actions can be used to define functionality (behavior) provided by the block like operations. This allows them to be re-used across multiple activity diagrams (this is different to call operations) and maintained more easily. Actions provide a much richer syntax then operations thus allowing us to describe more complex functionality in a more abstract way. For example they allow multiple inputs and outputs with continuous or discrete data flows, etc. 5. Requirement decomposition - Requirements should no longer be simple sentences. They should be decomposed into requirement pieces, e.g. model element satisfying it, operative part, shall or may, limiting parameters, etc. Simple text statements are more difficult to maintain since some of these parameters are associated with other modeling element. In addition it forces consistency in requirement variables and a measurable means of ensuring completeness. It also allows the tools to more easily search the model. 6. DeriveReq Relationship – Consider changing the stereotype named “DeriveReq” to have the verb phrase more intuitively agree with the direction of the arrow. Suggestions inlcude “DeriveFrom” or “DeriveFromReq”. A shorter name would be better, but this will avoid errors. 7. Description Property – Although the tools generally provide a description field it doesn’t appear to be included in the SysML specification. From a practical sense, it is important to provide a definition for model elements. This description provides another level of clarity, identifying what something is and what it is not. 8. Binding to Constraint Parameters – The language should allow tags and part attributes to be used to bind to constraint parameters. Tags are often used to capture values that become variables to the constraint equations. Part attributes contain the initial value and the part value. The block attribute can be used to bind the initial value, but the part attribute also can be used as a variable. 9. Pins and Flow Ports – A formal mechanism is needed to measure the constancy between data flowing on pins that cross swim lanes and the data defined within the corresponding flow ports. 10. Flow Port consistency – A more formal mechanism is needed to ensure consistency between ports, both from parts on the same level of composition and on different levels of composition. This allows the tool vendors to implement automated consistency checks within the tool. 11. Event Consistency – This may be more of a tool issue but provide a formal means of specifying events once and referencing these events across activity diagrams, state machines, and sequence diagrams. 12. Value Types – OMG should provide an importable profile (XMI) containing a comprehensive set of standard and useful value types with their defined value types and dimension specified. This is more than the set of SI units. This set should include the many value types that are used on a day to day bases including, ft/sec, m/sec, kilometers, bits, bytes, bits/sec, etc. etc. etc. Today everyone is creating them but everyone is doing it differently.

- From a language perspective, I would like to see activity diagrams being the same between

SysML and UML.

- A single interface block should be able to accommodate messages in both directions. Today, a standard port has to have a contract with one interface for provided or inbound messages and another for required or outbound. For data distribution centric systems, you should be able to have "data provided" (outbound) and "data required" (inbound) for a component where SysML currently only allows for SOA "provided services" (inbound) and required services (outbound).
- Tools should be required to produce QFD type reports, i.e., use cases-to-requirements and requirements-to-use cases, use cases-to-blocks, and vice versa, and blocks-to-requirements, and requirements-to-blocks. This would help ensure all requirements have been appropriately captured by at least one use case, and that each use case is traced to at least one block or interface (i.e., some object), and that at least one object satisfies the requirements. Currenty tools only allow a one way trace through the diagrams.
- Mechanism for defining more complex interfaces similar to those found in traditional ICDs, More guidance and examples for determining when to use flow ports vs standard ports. Sometimes vendor recommendations conflict such as the message based Harmony methodology with uses standard ports to provide context to data.
- Flow Port to Standard Port Interface Mapping
- Per discussion with Sandy Friedenthal related to sequence diagrams, suggest that a Pattern SME be identified and/or a reqt be added to the RFI/RFP to address Patterns. What are the fundamental reqts for implementing Patterns in the SysML diagrams, probably sequence diagrams but also all other related diagrams (activities, etc.). Suggest trying to implement a well know pattern or a few representative types, to determine what changes may be appropriate to SysML diagram types. Then can work towards reuse of these patterns via a library of these and any others that can be available for others SysML users to apply.
- We really needed an elegant and rich way to represent message based ICD flows in an aggregatable, hierarchical way while connecting those elements to sequence diagram flows and IBD ports.

Question 15: If you identified any issues with the diagram types in question 5, can you propose a solution to the problem? Note: Additional space is provided at the end of the survey in question 61 if needed.

Activity diagram

- Semantic specifications
- Allocating activities as operations to blocks
- Drag and drop an actor, block, "instance" onto the diagram creates a swimlane (just an alternate expression of the object on this view)
- Formal syntax and semantics e.g. based on OCL but with an easier syntax; oppinion based on C syntax. OCL standard syntax is hard to understand for people without mathematical or CS background
- Identify a subset that has a clear execution semantics that can be simulated correctly. The more complex constructs should be considered as "advanced"
- Clarify and improve handling of part/port names for multiplicity higher than 1 and consistency between BDDs and IBDs
- Define items currently not defined for use and define instance properties.
- Weight on outgoing vertices (from action to bar) should indicate token generation. I.e., implement standard Petri-net semantics.
- Clarify through examples.
- Association of tokens with parameters? some specialization so that flow is more like causality.
- Remove the ambiguity between pins and control objects.
- Actions as classifiers
- General: Provde more detail descriptons of SysML terms to avoid variations in usage and encourage model reuse for other projects.
- The SysML standard should specify the syntax and semantics of making activity diagrams executable
- Inheriting activity/action description directly into an requirement block for translation into a well written requirement (or requirements)
- Consider merging the capabilities of an action diagram into an activity diagram and drop the action diagram. Also make the decomposition of the activity diagram easier.

Block definition diagram

- Abstract block or pattern block
- Make sure connectors are not available. For any association on this diagram, always add a default label to the association line, so folks don't get the line confused with a connection. Or consider a different line type/style for any relational association vs. a physical/ logical connection.
- Look at how locally defined types are implemented in Modelica -- SysML needs a similar mechanism
- Similar to the proposal for ACD, identify the "basic", "intermediate" and "advanced" constructs of BDD.
- Add another compartment to block that list the ports / Clarify the usage of containment (e.g. for the structural compartment) and composition (e.g. for ibd)
- A mechanism to define special types of blocks, properties, operations, ports, connecters,etc.
- Clarify through examples.
- The term "part" is confusing, it would be better labled as an "instance" of a block.
- Need to treat messages as objects that can be expanded upon a lower levels

Internal block diagram

- New layout
- Allow selection of expression of the object on the diagram. Sometimes, I want to see the whole box; sometimes, I'd rather have that box appear as just a node (minor player on the diagram, like a translator between protocols, or providing a standard service), or express the object as a line (like a connector - pipe, wire, etc)
- See Q61
- Same as BDD
- Define junction ports that are virtual reference points of ports/parts within the block / Define the semantics of port types that own ports (=nested ports)
- Define items currently not defined for use and define instance properties.
- Clarify through examples.
- Change the Port notation to be more clearer
- I don' t believe a fix is required, but I believe this feature of the language needs to be better understood. Its subtle.
- Functional Block structures for highly abstracted views of a system
- Ref Sequence diagram CONOP/OpsCon comments.

Package diagram

- Eliminate, use only bdd & ibd for these purposes.: 1
- I discovered that creating a "Containment" relationship between an object and the package didn't really do anything. The object doesn't actually get move to the package. It was the only practical idea that I saw for this diagram, but didn't work. I would LOVE the idea of objects that don't actually live in any one package, but the packages are a means for me to present objects in meaningful ways, such that an object (one item) can appear in several different packages. Such that, in the project browser, I can create an object within one package (doesn't necessarily mean that is where it lives in the information model mind you), and then take that same object and create a Containment relationship for that object to another package, which then allows me to navigate to that same object through two (or more) different packages. Currently, the "Containment" doesn't even work. if I create a Package diagram, and drag an item already contained in that package onto the diagram, the Containment relationship between the Package and the items doesn't even appear!
- We propose to remove this diagram or make it non-normative
- Suggest looking into Database reqts for systems, etc. what package diagram relate specifically to DBs and are there needed changes to SysML.

Parametric diagram

- Eliminate, use only bdd & ibd for these purposes.: 1
- Rather than illustrative examples of it usage, a broad range of examples of its usage including where it would fit in the overall process would be very useful (at least to me).
- Adopt Modelica for semantics
- Define a simple and intuitive constraint language
- Give an example of nested constraints.
- Define items currently not defined for use and define instance properties.
- No suggestions, it just seems like these diagrams are of little practical value. Maybe remove them from the spec and let better suited tools like SIMULINK handle this sort of thing.
- Suggest looking into Database requirements for systems, etc. what parametrics relate specifically to DBs and are there needed changes to SysML.

Requirement diagram

- Eliminate, use only bdd & ibd for these purposes.: 1
- Perhaps OMG recommended constraints? or best practices guide?
- CASE tools integration modules
- Tabular views -tabular views. (You could replace DOORS with this one view!) Also add columns to tabular views to add relationship types (how does this item in row #20 relate to whom with a Containment relationship". Drag and drop relationship building (who needs a requirements diagram?)
- Make requirements derive from class/blocks so that they can have properties
- Implement Property-Based requirements support according to the theory proposed by Micouin (cf. point #61)
- Define a structured text for requirements that could reference other model elements.

- Add interactive tabular views, also supporting trace, allocate, derive etc. relationships to system decompositions
- Add parameters like constraints have
- Make them like constraints.
- This is a tool issue. I believe a good tool will provide a way to view all requirements in the package so that it is easier to manage.

Sequence diagram

- See notes above regarding cohesion with activity views and removing software semantic expression.
- There should be a kind of message for flow properties in sequence diagrams
- Similar to activity diagrams
- See Q61
- Add item flows to the interaction model or clarify that sequence diagrams are only for message based communication.
- Clarify through examples.
- Inheriting messages into interface classes as part of analysis
- Maybe consider adding sequence diagram data flows to the spec constraints in how they relate to the rest of the model.
- Need to identify a Pattern SME to determine the reqts for Patterns for sequence diagrams and any other associated ones (e.g. activity). Also review how sequence diagrams are applied by customers and during needs & reqts analysis when developing CONOPs/OpsCon for customers based on Operational analysis. Include in RFI/RFP.

State machine diagram

- MD
- Similar to activity diagrams
- Include protocol state machines to SysML.
- Define items currently not defined for use and define instance properties.
- Clarify through examples.
- Allowing parameterization of the states and display of those parameters. nuf said.
- States as model elements.

Use case diagram

- No
- This one revealed to me the "diagram-centric" thinking being promoted in SysML. Consider - if you pull any object from any other diagram onto a different diagram, how should it be presented? Actors as stick figures on this diagram are good, but stick figure on the requirements diagram (used to show stakeholder expressing a need), swimlane on the activity diagram (used to show participation in the scenario), block or stick figure on the BDD/IBD - to show "kind of" or interfaces with other blocks. So, for any object from any one diagram, how might that object be used/expressed on another diagram?
- Provide guidelines for the usage of use cases
- Add a decomposable actor, i.e. a special kind of a block that is external to the system.
- Investigate other modeling tools and/or provide in-house training
- Allow the user to select and associate multiple higher level requirements for analysis by use cases
- Ref Sequence diagram comments on scenario development for CONOPs/OpsCons.

Other features (e.g. allocations)

- Manage quality of methods and Attributes
- Reuse MARTE <<assign>> allocation
- refine definition of inheritance, interfaces, etc to remove software specific modeling
- SysML is really lacking in the definition of reports. The requirements tables are really great, more output like this would be useful
- As far as tables for allocations why does the standard specify this for depiction purposes? Why can't I enter allocations between element entities in a table?
- Allow ports to be initially defined at a high level model, then reused into lower decompositions

Question 16: Identify SysML specification changes you recommend and why?

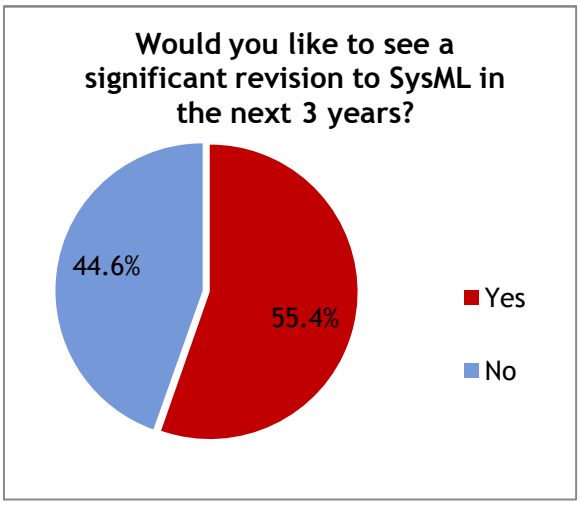
- The SysML was originally developed to minimize the complexity of using the UML in systems engineering. In doing so, the language is slightly more complicated than necessary. The flow concept between ports should be extended to include non-flow static pressures and potential differences. (We use bond graphs to model these for simulation).
- Eliminate the <<block>> as the basic stereotype. Revert back to <<class>>.
- I am not sure why they did away with the concept of association classes for relationships. Context diagrams don't seem to be explicit enough. Create model extensions (templates/stencils) for other conventional modeling domains (e.g. digital or analog electronics)
- Remove the entire Requirements portion and make it a stand-alone OMG specification. it is applicable across all domains and not just SysML.
- Clearly specify block instances and provide a clear graphical representation
- Splatterted up above to make job easier. Need Architecture Modeling also
- SysML must be extended temporally to underdetermined and complex dynamical systems.
- Eliminate redundant diagrams & streamline the language.: 1 Not familiar enough with the spec to advise..... we work from alternate SysML training sources.: 1
- Abstracts blocks or Pattern blocks Simplify internal blocks Improve requirements managment Semantic for the contexts The reasons for this changes can be seen in the previous questions.
- Adding SOA constructs to the SysML specification.
- I have a copy, and find it very hard to follow. Certain words have very precise semantic meanings. Sometimes, I've found it to be confusing, and just plain odd. The example is the "property" to express a "part". Very difficult to follow, since words are being used in a atypical sense of the words. Mostly comments relate back to the "information model" vs. the "diagrams". The standard needs to focus more on the proper relationships between different types of model objects, and focus on the diagrams as views of those different modeling objects.
- Need definition of a simple "default" methodology.
- The SysML specification is too much diagram oriented. Diagrams are only views of an underlying model. The description of the semantic of SysML concepts should be more formal. A domain model would be useful to clarify and consolidate some of them.
- To define the architectural context for a system always block instances should be used. The usage of blocks with embedded block properties is not practicable, because the system context can change. Furthermore an instance of a block used as system context is always the same. But this can differ in variants. In our model only instances of blocks (properties) and internal block diagrams are used to define architecture. The blocks are used as a kind of metamodel defining the domain specific terms.
- Remove UML4SysML concept. It artificially creates many problems and limitations. It makes no sense, as there is no "pure SysML" tool, all implementations are based on UML tools. harmonize MARTE and SysML

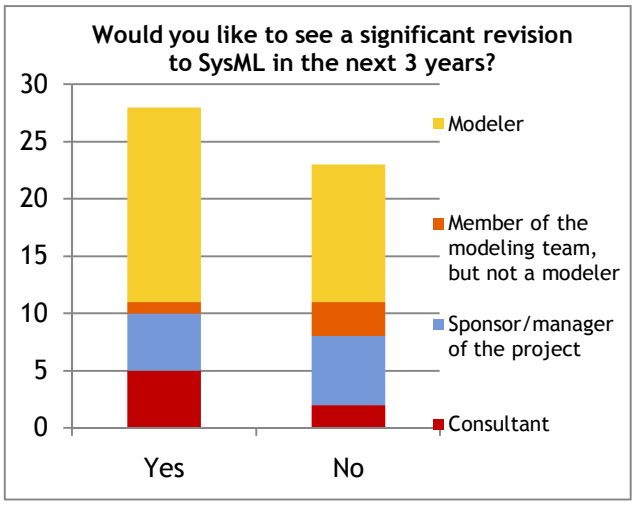
- In addition to the items mentioned wrt to question 14, we propose: 1. Support the notion of part specification and not rely on nested connector ends – this will simplify tool implementation to reference properties in context of a part hierarchy 2. Future versions of SysML should be designed so it is easier to integrate related profiles and modeling languages like BPMN and SoaML 3. Identify or define packages in the meta-model for basic, intermediate and advanced so beginners can start with the basic concepts and then move on to the more advanced elements as they gain experience with using the language 4. Give more emphasize on guidelines and "best practices" to explain the intended usage of the various diagrams
- Split SysML specification more and more clearly than is currently the case (v1.1 and v1.2) into a formal language normative part and a "handbook" / "guidelines" / "best practices". Currently the normative parts are not sufficiently rigorous and precise. Also the guidelines are too much interspersed with the formal language specifications. Consider a Light SysML specification based on Eclipse/Ecore, to help create advanced and flexible tools (COTS and free open source), and integration / data interchange with other non-SysML tools.
- 1 is low importance and 5 is high importance 5 - More support for variant modeling 5 - Nested ports 4 - Better integration of activity and action in parametrics diagrams 4 - More support for multilayer modeling and multidimensional aspects 3 - Stereotypes and icons for mechanical, electrical, information, optical, thermal interfaces, represented by ports in the non-normative section 3 - Harmonize UML profile for QoS, and UML profile for Real-time (MARTE).
- To incorporate a mechanism to define domain-specific models because each domain already has a domain-specific model to express systems in their domain.
- Describe the SysML as a metamodel and a corresponding UML profile. Some tools might implement "natively" the SysML metamodel but not the UML metamodel with a SysML profile on top. UML tools with a SysML profile have to do a lot of SysML-specific customization to make the tool "feel" like it is a native SysML modeling tool even though it isn't. This is a noble effort but it is a lot of work. It would be much better if we could simplify the problem using Metamodel-to-Metamodel transformations in QVT and let tool vendors focus on implementing a given metamodel properly without split-personalities between UML and SysML.
- Add SysML metemodel
- I was my understanding that the end product of the Architecture design was going to produce a framework for the developers. However, it was difficult to illustrate simple logic (if, for, while) in block diagrams. Maybe SysML can bring that detail level of design up to architecture so that simple logic can be captured as opposed to reading a mini-spec.
- Form based definition for each model element would assist in completion of the models from the top level. Inheritance of scenario attributes from higher levels might also highlight lacking models at the lower levels where higher level models have not been developed.
- The systems engineering analysis the supports the UML/SysML architecture and specification should be provided.
- I would recommend limiting or postponing changes to allow tool vendors to achieve functional and complete implementations that can be strenuously exercised on programs.

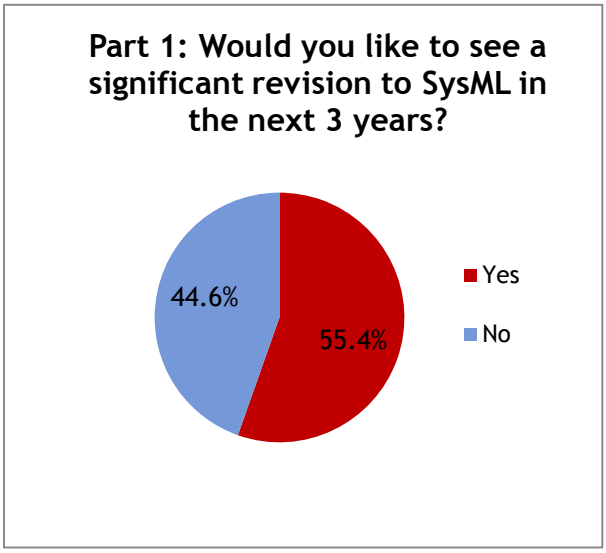
- I would like to see more output like requirements tables. As an example, a table for interfaces would be really useful. Its just too hard to get anything out of the SysML tools in a useful format to systems engineers. We still live in a world that needs to produce documents that can be read by someone that has no idea they were produced in a model.
- 0. More explicit identification of UML features not allowed (and why). 1. Supply better and more consistent examples 2. Perform an internal traceability check to prevent the continual finding of errors a. Every language feature should be illustrated in at least one example b. Every multiplicity limit in text should be tied to a metamodel diagram c. Every prohibition in the text should be traceable to an OCL statement (that has been checked) 3. Provide a standard notational hand-out covering the features used 90% of the time 4. Provide index with references to diagrams defining or illustrating feature
- I recommend that for the structural diagrams such as Class and Package to be use there needs to be cohesion and coupling metrics associated with them to be able to quantity how one structure is better then another. There needs to be tools that calculate these metrics. Otherwise they are just pretty diagrams with very little meaning.
- Require standard, parsable model and algorithm definitions.
- Expansion of the capability to link DoDAF models to SysML models
- Allow instances to achieve a better communication between systems and software.
- Insufficient experience to make useful recommendations
- See 15 activities/actions recommendation. Also, With the push from DoDAF 2.0 and UPDM on information and data (DM2, etc.) recommend reviewing SysML diagrams to focus on what reqts, parametrics, package, bdd, ibd, etc. are needed to develop the interfaces, components, parametric and reqts needed for Databases. Need to identify a DB SME(s) and a pilot effort to identify what additional diagram information is needed during system developments for DBs. This is a specialized field but extremely important in today and future systems.

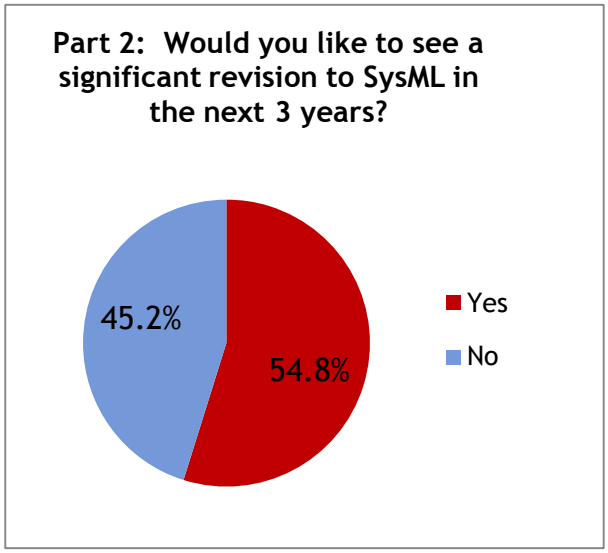
Question 17: Would you like to see a significant revision to SysML in the next 3 years? Please elaborate on your answer.

Results









Open Ended Responses

- But it doesn't really matter. Whatever limitations exist within the language are easily overcome. The SysML as it exists is VERY underutilized from what I have seen. Any changes should better the underlying mathematical graph formalism - which is ignored by most modelers.
- I cannot answer this without a definition for "significant".

- Need to stabilize the Profile and let the users get use to working with it. Minor revisions are fine but let's work with it for a couple of years before we take a major revision.
- It doesn't feel seamless and straightforward, it may be the implementation in the tools.
- A significant revision will cause a large ripple effect on the tool vendors and the current practitioners - as well as open the issue of 'upgrade' or 'compatibility' issues. It would be much better if the revisions were done in an incremental or evolutionary fashion to fix or tighten semantics, enhance readability or simplify notations.
- Let people enough time to master it as it is!
- And useable Tools
- Closer alignment to the original UML formalism (alignment thru UML4SysML) and less profile extension. Many of the changes I've noted actually need to be made in the UML superstructure (or even infrastructure) levels.
- No: 3 yes: 2 Better to make a significant change to SysML sooner rather than later. Keep it simple, yet vibrant.: 1 Don't make it to big and thereby limit folks willingness to use..... Its a good tool... lets just go use it.: 1 slight modifications, more focus on hardware design: 1 SysML should provide a clear path the SW design language : 1 We are still in the process of learning and implementing SysML. Significant changes will derail efforts.: 1
- Would rather see it evolve, unless that is impractical.
- Let people catch up to what is there now.
- Just getting use to it. Like stability in the language.
- Needs to better fit the SOA environments.
- Comments already expressed above!
- The semantics need to be tightened, and there needs to be an opportunity to make adjustments based on the lessons learned from applying SysML to larger problems.
- Make it simpler better support of product family lines
- UML will evolve and then SysML will have to evolve too.
- We are not sure with our answer, because SysML allows extensions by using the profile mechanism.
- Harmonize MARTE and SysML
- We need to improve SysML if we want it to gain acceptance in the industry, currently the level of acceptance is still too low
- SysML is used in many projects, but it lacks some important features. The projects define their own extension of SysML to fill the gap. It is time to provide a standard for these extensions in the next 3 years.
- In next two years SysML should proof its usefulness based on the current state, with only small enhancements. Next real iteration should probably be prepared to run in 2011/2012.
- I think SysML will need to mature rapidly in terms of the basics of systems modeling. Systems models are arbitrarily abstract so the language needs to be simpler and more capable than UML which has the specific products of software and source code as a foundation. Capturing the useful part of UML and discarding the rest in favor of whats useful to Systems Engineers will be critical for the integrity of the language.

- No, I would refrain from major changes. The base is very good, in particular compared to the first UML versions. There has been a lot of work invested the past 2 years to find acceptance of the language and carrying out major changes would invalidate parts of this work. Certain areas need to be addressed (e.g. variant modeling) and others clarified or fine tuned(e.g. nested ports, attributes for requirements). Not allowing nested ports would be for me a major drawback which would invalidate a lot of work. Some areas need to be consolidated pretty soon before the current, sometimes different, interpretations of the spec establish in daily practice and SysML usage diverges. Too significant revisions have the danger that people have to forget or redo the things they have learned and applied over years. Non software people are more sensitive to frequent or significant changes and makes them feel that the have lost their time. It is much harder to accept change for System engineers. Signifcant changes would also give a picture of a language which is made by SW-engineers, as "they always change what they have just released". Backwards compatibility should be on of the major drivers. However the identified issues should be resolved and the additional capabilities should be addressed within the next 3 years.
- It would be highly appreciated if a mechanism to define domain-specific models were incorporated.
- If you consider making things even simpler and more elegant a significant revision, yes
- Anything that can be done to simplify the language should be considered. I believe SysML will only gain acceptance when it is more easily understood and applied.
- A significant effort to improve SysML w.r.t. some of the problems and/or suggestions offered above.
- Among other things, to address the knowledge representation integration in item 13 above.
- Yes - but not a bloating. SysML has virtually all the ingredients. it now just needs to be consolidated in such a way that its full power can be unleashed.
- I would like to see the language semantics include a more rigorous mathematical formalism, increased capability for integration with other modeling languages, easier to use with enhanced graphical representations with more interesting iconic representations.
- I believe this language has great potential but needs to capitalize on the fact that a lot of people are interested in it now, not later. Time is of the essence here.
- Our experience level with the spec is too low to advocate significant changes at this time.
- It would, of course, depend on what revisions would take place. From the limited exposure I have had, I do think Lockheed is doing a great job using SysML and it is being implemented effectively.
- I don't work with architecture design anymore so I am rather I agnostic
- Mainly because when allowed to go stale, other adaptation or migration takes place. Continued language development is essential.
- UML and SysML appear to be to complicated. A simpler generic modeling language that is designed to be extendable with extensions that then exist for SW design, SW code generation, SE requirements analysis, SE requirements management, SE architecture, and System I&T. The common generic language should consist of the functionality to allow all extensions to be created and integrated.
- See 16
- Actions as classifiers!

- A simple circuit diagram or mechanical system with springs and dampers can easily be drawn in a way that everyone understands. I think a more intuitive graphical interface would help SysML gain acceptance in the SE community as would a standardized approach to getting word documents out of the tools.
- Change to make tool usage easier.
- Support for formal language export: Z or SCR. Support for automated formal verification: FX and TAF
- As long as it is mostly compatible with current version
- Make interface definition (with messages) more robust. Find a way to incorporate design in a way that allows handoff to SW AND HW. How can flow ports and standard ports be used in conjunction at both they system level and the software level?
- No comment
- Anything that simplifies the notation and reduces its scope would be a good idea.
- I'm guessing making the model more executable would require substantial updates to the standard.
- Reverse engineering of a system. Basically, I need to perform an “autopsy” on a dead or compromised system to understand why it isn’t performing as expected.
- SysML suits our needs (for now).
- increased standard library of interfaces for analysis
- Better I/F to DoDAF which is the model of choice for our DoD customers
- I think the present language is an excellent start. Any changes must be as backward compatible as possible and if not a reasonable transition needs to be defined. We have a significant investment in the existing path and we don't want to lose it.
- Too early to tell.
- Starting to integrate SysML into group.
- I say no because I have just not had enough time to work with the current revision.
- Tool vendors need to address interoperability. A significant revision will create further dependency on vendors that support the new revision and less focus on interoperability
- The language definition is good - it's the tool (e.g. Rhapsody) that is just barely ready for widespread use.
- Although I used SysML on several R&D projects, none of them are for a contract so the use has been limited in time and scope.
- There is a significant learning curve associated with the standard, and it would be helpful to many of us to have time to learn and understand the nuances of the existing standard before forging ahead with a new and improved standard.
- Just solid reaction to the changing environments we work in. But I do not want a significant change until the need for change or a real solution (like the business items I stated). Impacts are one concern but non-real solutions are dangerous to a successful language.
- It's easy to say increase ability to model highly dynamic systems than to implement
- Only if needed to keep up with Systems Engineering concepts and standards.
- I think a lot of usage issues about SysML could be resolved if we see more compliance to the existing spec first.
- Just need the tools to more closely implement the specification.

- Cannot answer this yet but I assume that usage of SysML and the evolution of the tools quality will lead to improvements and fixes as they are identified.
- Yes, no, not sure if our biggest issues (messages, etc.) are due to specification issues, or tool implementation of specification. Leaning towards tool implementation as the culprit.
- The language is good, learning it and getting best practices in place needs to happen

Question 18: If we do make a significant revision, what changes are most critical to enhance adoption of SysML and MBSE?

- Personally, I think allowing the easy simulation of models for verification and validation is key to adoption of SysML and MBSE. Some confuse this with "executable SysML" which I think hints more at a software-centric systems viewpoint. In simulation, I have hinted above at the separation of concerns between system models and their contexts. One of these contexts is the spatio-temporal domain of time, distance and frequency. All of this is a gentle criticism of systems engineering as an exercise of Archimedian statics, an inadequate paradigm in my opinion. But the SE domain is far bigger than one systems scientist in ABQ
- Ability to economically create electronic presentations for customers of the information (read only), and to update those presentations as needed.
- Better integration with the Testing Community (Developmental and Operational Test) Better integration with Product Life Cycle (PLC) Reliability and Maintainability.
- Seamless integration with standard plug-ins, e.g. stochastic tools, etc. Most systems are not as deterministic as SysML imposes.
- More explicit semantics and better tooling
- Multi-model hierarchy support, support for using sequence diagrams for testing.
- UML Timing and Communication diagrams Block instances
- Advices: 1) maintain SysML independence from the domain. 2) consequently, do not try to replace domain specific notations with SysML.
- Architecture Tools. How do Objects inter-relate
- The ability to test the model hypotheses with real-world constructs through data.
- Action semantics, linkage to analysis, proven profiles linking to SW & PM: 1 An MBSE process would be good -but that might be a negative thing as the language should be independent of process?: 1 Tools, training, licenses: 1
- If it could be released with elaborate examples (though perhaps narrow ins scope) of applications that may serve as templates for programs and projects to adapt, versus develop from scratch. E.g., a weapon system, versus a pulley system. The best way for me to learn and re-learn software languages is to use some of my old ones or someone else's as starting points.
- Flow ports need to be simplified so that they do not send overly complex types. Stop mixing Activity diagrams and State Machine Diagrams
- The fourth of the question 16 answer
- Testing and Security.
- Making SysML a language where engineers could choose either modeling with traditional constructs or OO constructs.
- Information Model focus (5) - objects and relationships Connectors treated as objects - pipes, wires, etc (5) Abstract Layering concepts (5) Object Configuration Control (5) - concepts to place objects under config control, not just whole models or packages, but an underlying configuration mgt structure to check-in/check-out individual objects Tabular Views
- Move SW and SE modeling together so SW engineering adoption doesn't have barriers. This results in either lack of MBSE or SysML use or duplication of some work.
- Be more formal for both the semantics and the concrete syntax. Add a domain model for each SyML concept (cf. MARTE specification). This will help in improving the SysML specification.

- Integration of the UML testing profile - Formal syntax and semantics for behavior modeling - Modeling capability for timing behavior
- Harmonize MARTE and SysML
- As mentioned above: make it even simpler for non-MBSE practitioners to learn and use.
- It is important to extend SysML, but not to change the current "look and feel". The market won't accept a significant revision that requires new tools, new trainings, new concepts, new methodology, and so on.
- Systematic feedback from actual usage. - Documented integration with one or more MBSE methodologies. - High quality model libraries for commonly used concepts: quantities, units, - Simplified Ecore based meta-model. - APIs for connections with other tools. - Better support and more precise semantics for system decompositions (block/part/port/connector hierarchies), value properties (Quantities, Units, Dimensions and Values) - Property based requirements - Standard tabular views in addition to existing diagrams
- Include ALL stakeholders clearly. There are too many arguments concerning who/what should be included in Use Cases. My position is that anyone or anything affected by the system being designed and developed should in some manner be represented in Use Cases and their associated requirements.
- Adding instance values to IBDs, activities and states is absolutely critical. Tables of some form for all diagrams are critical to.
- Variant modeling and property specific types - Nested ports - Integration of requirements and activity attributes in parametrics - Discipline specific stereotypes and icons, e.g. to model mechanical, electrical, optical interfaces - Change some of the software-related terminology, e.g. interfaces - Junction ports - Show internal structure of actions - as done in IBDs
- Support of assumptions and promises is an area where we see a lot of promise for improving efficiency in Systems Engineering. Graphical Distinction between physical and information/data flow Version, variant and configuration management
- Incorporation of a mechanism to define domain-specific models.
- I'm excited about the possibilities of SysML, but it feels like I'm trying to model the system in Assembly (metaphorically) rather than Python. The effort to properly structure the model and represent common SE concepts is immense. Perhaps this is a tool / profile issue rather than a language problem, but I would rather be dealing with language constructs that have the same level of abstraction as the SE process, say IEEE 1220-2005, or the DoD's SE Fundamentals, rather than have to recreate those concepts. I would prefer the UML/MOF heritage to be hidden behind a more familiar API.
- Same as #17...Anything that can be done to simplify the language should be considered. I believe SysML will only gain acceptance when it is more easily understood and applied.

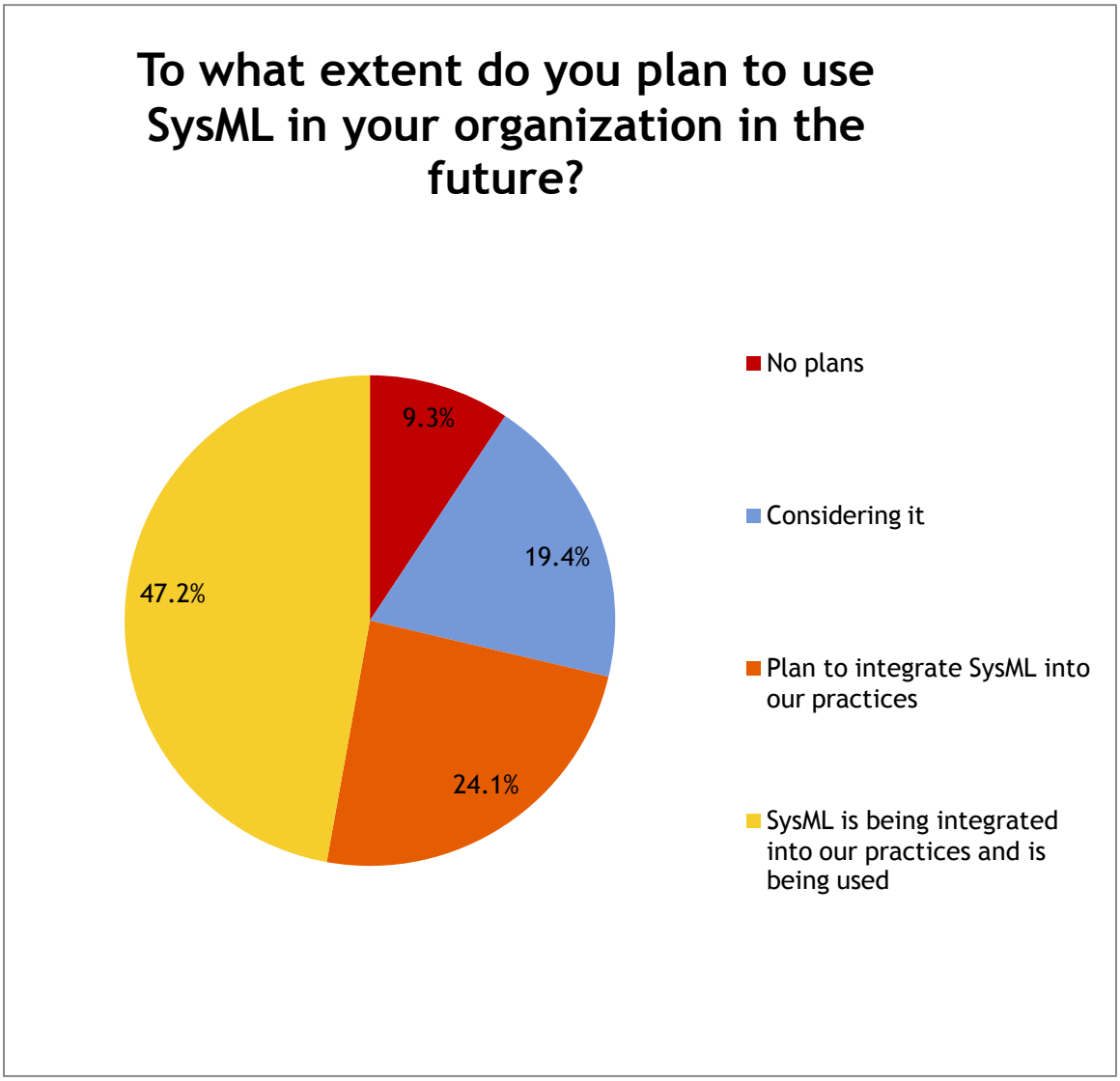
- Overall: The scope of the spec needs to think beyond diagram notation to include factors that help SysML fit better in the overall MBSE enterprise (especially re: meta-data that will enable better collaboration). These factors include the following (many of which need to be part of the model but not necessary part of a graphical diagram—in other words, the spec should be focused on model content, whether it be graphical or not, and consider the diagrams to be useful windows/views for working with subsets of that content): a) Configuration mgt. and version control, both within models and across models and specs. Here are a few examples ... i) As SysML-enabled MBSE environments grow and mature, they will need to support concurrent usage of multiple versions of the SysML spec, and even allow a SysML model to have sub-portions that are based on different spec versions. ii) The same environment will need to support multiple versions of the same model library, as Project A may use Library v2.0, while Project B is at the same time using Library v2.1 b) Robust security and access control mechanisms will become essential. The drivers behind these are often naturally part of the model meta-data and include contractual and intellectual property considerations. Some tools may support some of these aspects, but they need to become part of the model. Related capabilities include: i) Ability to let someone use a block and see its interface, but not to see inside that block. ii) Ability to use and see inside a block, but not be able to change it. This should be more enduring than simply locking for edit (which can come and go). c) Support for model interoperability (not just XMI-based file exchange). This would include supporting an organization that has SysML modeling tools from several different vendors such that User 1 with Tool A and User 2 with Tool B can work on the same model in a shared repository.
- QVT support. QVT is really a paradigm-changing specification because it has wide implications for several things that MBSE-proponents are asking for: - model interchange. Instead of doing interchange at the XMI level, we should have interchange at the level of QVT support. That is, tools should provide the same results for the same models & the same queries or the same transformations. - QVT Operational, a better OCL. In practice, current tool support for OCL in UML/SysML languages has been and still is pathetic. With a good QVT implementation, who needs an implementation language? The implementation of QVT Operational in the Eclipse M2M project is pretty good; it includes support for the best OCL implementation bar none. http://www.eclipse.org/m2m/ I used a very preliminary version of QVT Operational to write all of the OCL in the QUDV conceptual model, including the specification for the coherence checking and dimension analysis algorithms.
- Its gotta be streamlined so its easier to learn. and I don't mean easier to learn like systems engineering will ever be easier to learn. rather that once you start to get your rhythm with it - there is all this extra stuff to keep track of like differences between act and seq or extracting manually the interfaces implied by swimlanes on activity diagrams.
- Executable activities with timelines that are tightly integrated with the structure views. Enhanced parametric capability to integrate with engineering analysis and simulation. Much of this must be done by tool vendors. More extensive model libraries.
- Timely support by existing tools already being used.
- Make it simpler. Look for ways to eliminate ambiguity, have a standard that is simpler to read (standards are indeed hard but the point here is making it simpler would foster adoption of it--that's why people are using the INCOSE 2006 presentation to understand the standard; the question then is why the standard was written if it was not to communicate the message clearly?).
- Linkage to requirements, DoDAF views.

- Similar to an IEEE guide that accompanies an IEEE std- a SysML guidebook to accompany the language definition would increase adoption.
- Good modeling and systems engineer requires lots of work. To get these efforts universally adopted there needs to be an easy and common way to reuse the work products. Architecture, Requirements, Analysis, and Models need to be reusable and extendable.
- The most critical element to the adoption of SysML and MBSE is to establish precedents for design critical artifacts using SysML and MBSE. Refine MBSE to be more clearly Product based identifying how the MBSE workflows align with, and provide, the form and function of traditional Systems Engineering products such as IRDs, functional schematics, ConOps, etc.
- Actions as classifiers!
- 1. Getting documents such as specifications out of the tools. 2. Improved graphical user interface (i.e. diagrams that look more like what an SE would draw on paper).
- Clear language definitions
- Support for formal language export: Z or SCR. Support for automated formal verification: FX and TAF
- Make interface definition (with messages) more robust. Find a way to incorporate design in a way that allows handoff to SW AND HW. How can flow ports and standard ports be used in conjunction at both they system level and the software level?
- no comment
- I don't think adoption is a good idea
- It needs to become the equivalent of XML in that it is the common language that all design tools talk to. At this point, SysML is just another island you need to build between where you are and where you need to be.
- Improved interface modeling
- Block Inheritance defined Requirement Mapping Actions as properties of blocks Requirement decomposition Binding to Constraint Parameters
- Improve "presentation" for non-SysML users... Not sure how to quantify this. We are somewhat constrained by the IBM Rhapsody tool we are forced to use, so that is an issue. The diagrams are usually too complicated to paste into PowerPoint and remain readable. Maybe due to our limited experience. Maybe a tool constraint.
- Although the SysML language, like UML, takes "pride" in being process independent, the cost is a higher learning curve. It is natural to have a process drive a notation, albeit a general process, therefore, I would like to see a suggested order of things such as the Rhapsody Harmony Process is used to create a flow given some basic understandings when selecting how best to use the language.
- XMI compliance and model interchangeability between vendors. Vendor differentiation should be based on ease of effectively creating SysML model that conform to interchangeability. Currently emphasis is on ability to create a SysML based model with little concern for interoperability.
- SysML effectively support SOA but it could use more support for data distribution centric systems designs.
- Usability -- there is a major challenge getting systems engineers to accept and effectively use the methodology and associated tools. Added complexity will hinder this institutional acceptance.

- Integration/Incorporation of the UML2 Testing Profile at the system level.
- Simplify the steep learning curve, if possible.
- Good examples.
- See 15. Also need to compare with DoDAF 2.0 & UPDM and work through examples of its implementation to determine where improvements are needed for each DoDAF 2.0/UPDM model/information.

Question 19: To what extent do you plan to use SysML in your organization in the future? Other (please specify).

Results

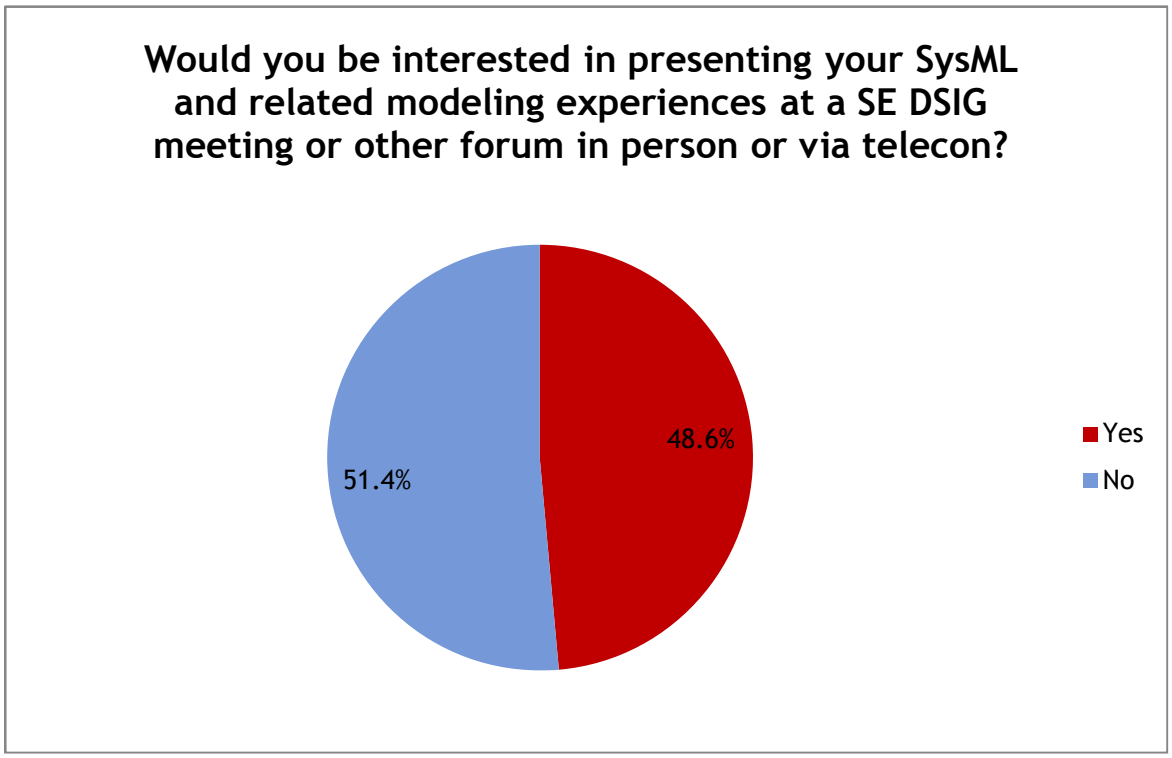


Open Ended Responses

- Shhhh. This is a secret plan.
- Assessing how it fits.
- I am a Consultant and Trainer in my organization
- Engineers interested. Managers so-so
- SysML is being integrated into our practices & is being used: 4 plan to integrate SysML into our practices: 1
- --the days of "practices" have come and gone in my govt. organization
- Mix of use and non-use. program dependent more than organization

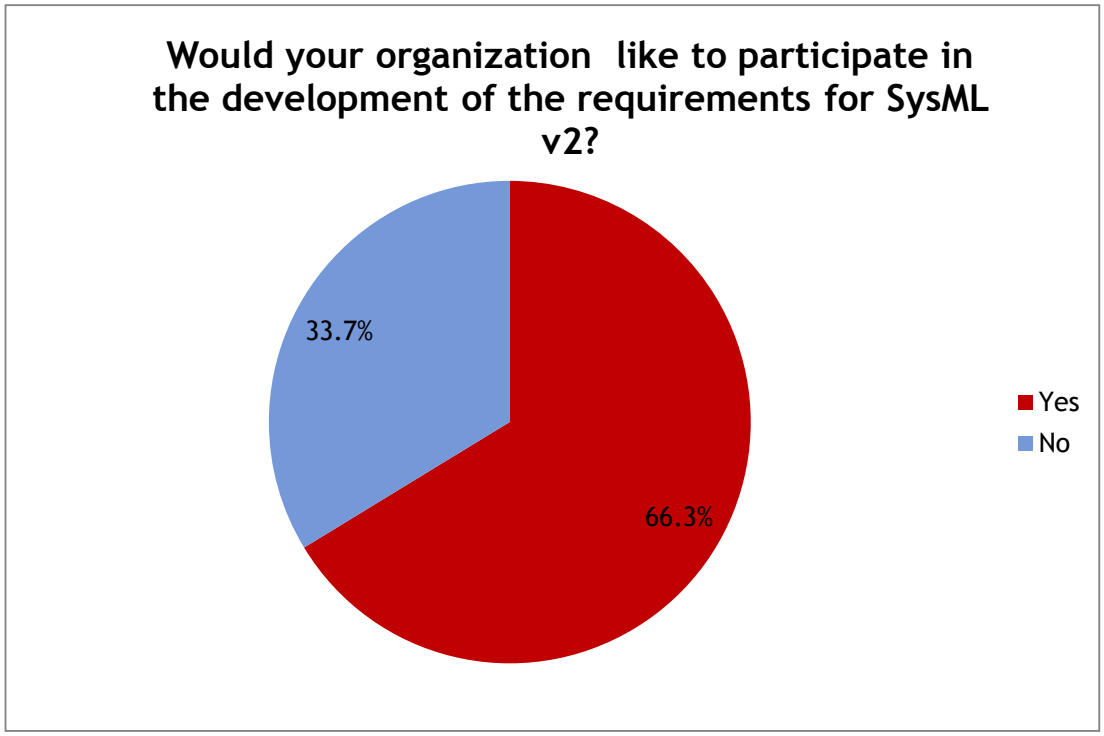
- SysML is an important part of my research program
- Undertaking pilot project to assess. TBD whether will be adopted widely.
- We are tool vendors that support and promote the SysML standard and we plan to continue doing so
- Currently restricted to R&D. Increased usage in real projects expected in coming years.
- Actively taught
- SysML is being used, however these critical issues are barriers to wider-spread adoption.
- Use is as required by customer
- Teach classes, lunch and learns, and spread the news around to generate awareness of it.
- I am no longer on a System Engineer team but while I was, we were actively using SysML practices
- The current environment only used UML when it is required by the customer.
- Although I like SysML, most of the SE community does not. It's difficult to convince others to use it.
- Unfortunately
- I am currently converting a document-based SE process into a Model-Based SE process by building a System Model with SysML.
- I don't have a say. I would use it on all projects at the system level.
- We have both advocacy and resistance co-existing. I believe the advocates will prevail in the long run, but there are many objections yet to be countered.
- Substantial resistance from the current engineering community
- Depends on my next position, if I get into the new enterprise architecture TTO, hope to work SysML implementation for DoDAF 1.5 and 2.0 into the Enterprise Architecture.
- Company needs to have regular training classes established to train Sys Engineers on this methodology

Question 20: Would you be interested in presenting your SysML and related modeling experiences at a SE DSIG meeting or other forum in person or via telecon?



Question 21: Would your organization like to participate in the development of the requirements for SysML v2?

Results

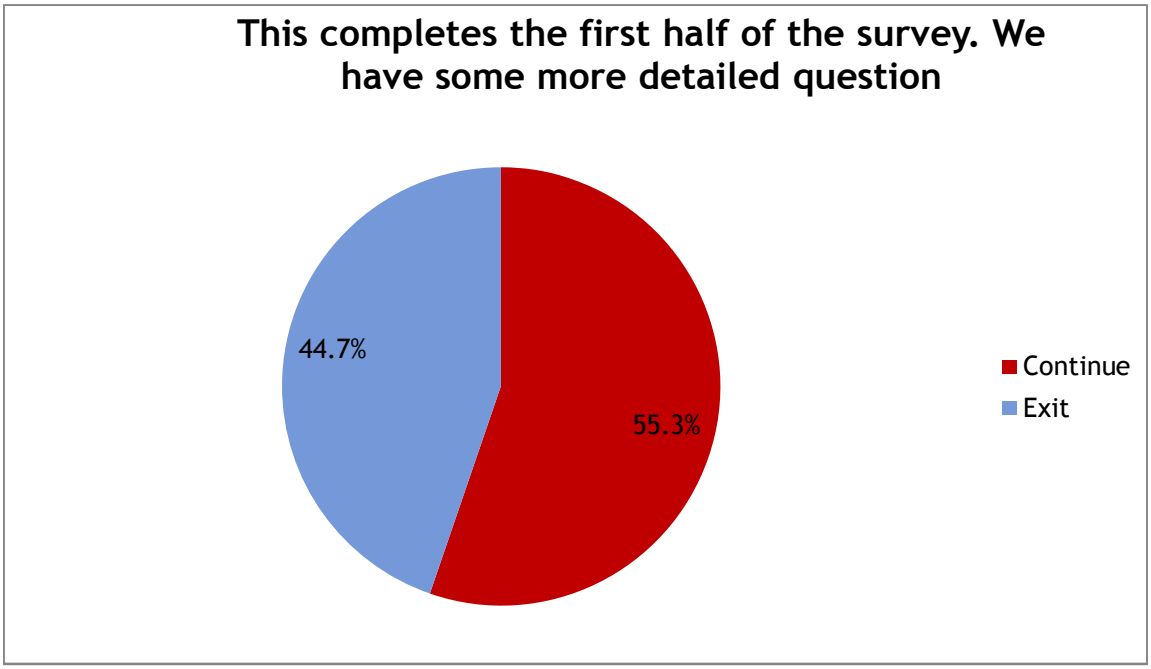


Open Ended Responses

- I might. The company at this time has no RADAR that will pick this up.
- About question 20: depends on the when and time required.
- My requirements do not seem to align with the general practice.
- Sort of - but need someone outside to put pressure to increase our involvement; otherwise, probably won't happen.
- Not sophisticated enough user yet to be able to be of use near term...
- Yes, but don't know the details.
- Not at this point, but maybe down the line if we start using it more
- I personally would like to. I cannot state that my organization would like to.
- We are participating.
- I won't be able to answer for the organization but I would be interested personally.
- I'm not in a position of leadership to decide that for the team.
- I can only speak for myself I have spent more time on SysML then I care to
- Don't know
- Depending on the willingness of the organization to fund this participation
- Maybe..I am not the person to determine this

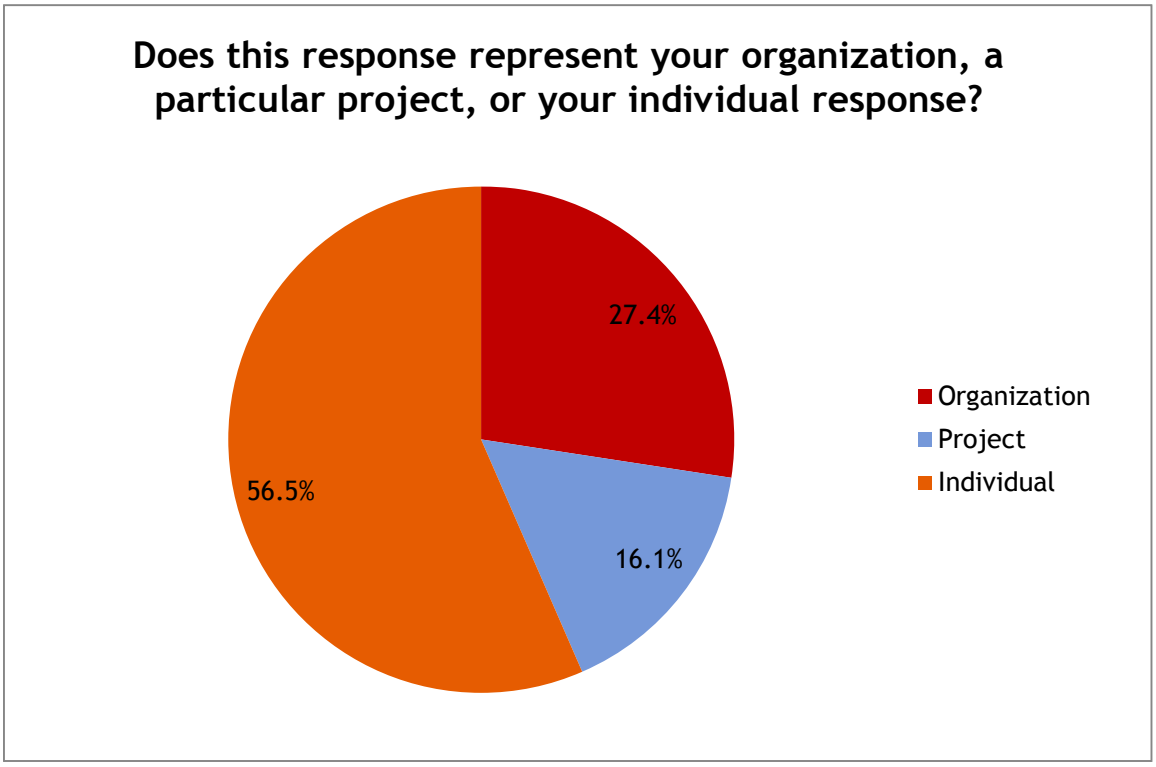
- Not sure of this really - but "No" is a strong word.
- Already does.
- I'm sure we have a few representatives from our org already
- Personnel are involved at this time.
- Personally, if I can get into developing SysML diagrams for my program, I believe I can contribute. Since I am also evaluating UPDM, I think I will have some higher level comments/suggestions.

Question 22: This completes the first half of the survey. We have some more detailed questions we would like to ask if you have time. Do you want to continue, or exit here (if you exit here, your answers to this point are recorded)? If you want to maintain a hard copy of this, please print this using your browser print function before proceeding.



Question 23: Does this response represent your organization, a particular project, or your individual response?

Results



Open Ended Responses

- I am the sole practitioner, and have used it on multiple projects.
- The Systems and Management Department represents AFIT
- A little of all three, but notes taken by myself. I represent out Enterprise team that has been assisting a project in application of SysML - with some success, but a lot of frustration to get there.
- Two business units in our company with currently two main development projects.
- Pilot project for organization
- We are a consultant company. Our response represent the experience from our customer many projects from different disciplines.
- However usage of SysML at ESA is still in its infancy so there is no wide usage or experience yet.
- INCOSE SSWG Challenge Team, and some internal pilots at JPL
- MBSE Challenge Team SE^2
- But I am the one person who brought SysML into my small organization
- There are a lot of folks at JPL using SysML; too many for me to contact and poll for an organization-wide consensus. I hope that others at JPL have taken the time to respond to the survey.

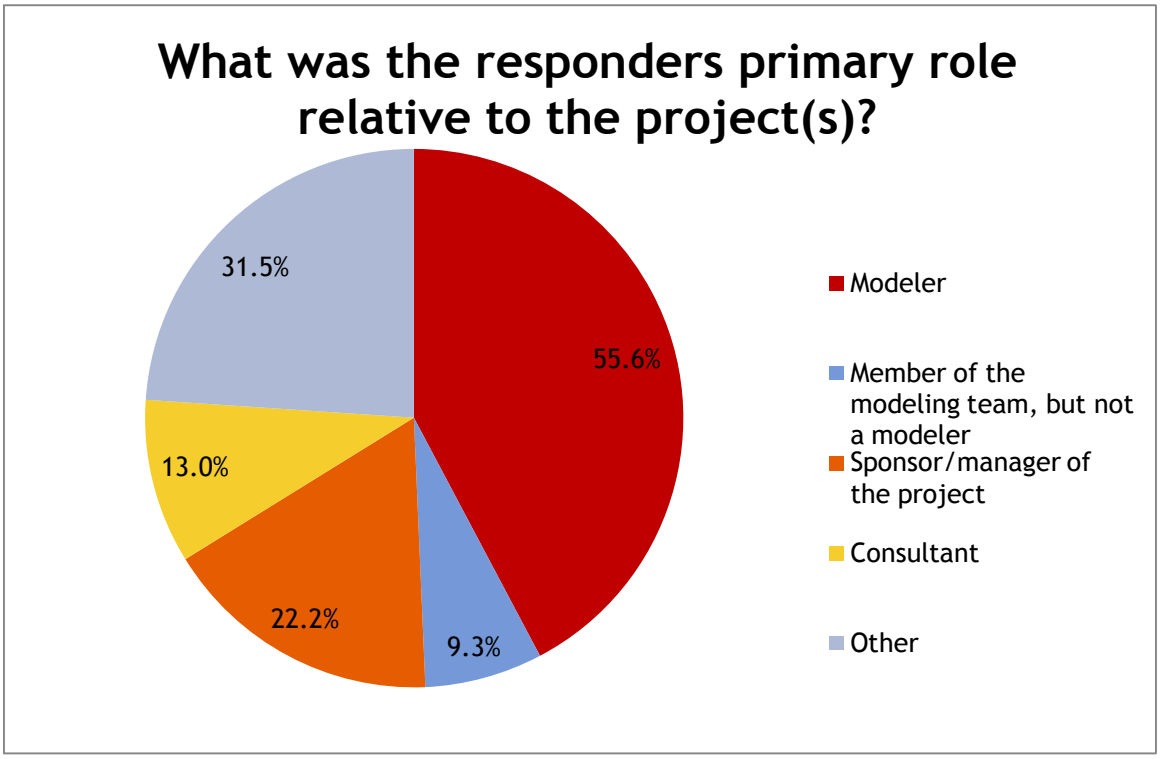
- This is the response from me and my lab (www.msl.gatech.edu), and other colleagues from GIT may response separately (e.g. Chris Paredis).
- A Project and individual response
- Individual experience on a project using SysML
- But it is a composite of issues discussed in the work place and some academic discussions

Question 24: If this survey is being answered from the point of view of a particular project, please enter some unique project identifier of your choosing.

- Mission Operations
- Military helicopter avionics project - hardware and software development
- Electronic Brake System Development
- RCM GS
- 2 R&D activities: Virtual Spacecraft Design (VSD) and System and Software Functional Requirements Techniques (SSFRT) to improve Systems/Software Co-engineering
- Space Systems
- Development of Spacecraft Information Base
- SOA
- Most of my opinions reflect the "Integrated Model-Centric Engineering" (IMCE) project at JPL albeit from the point of view of my experience working with OMG specifications for several years.
- It is regarding experience with multiple projects / organizations - see question 6 response.
- Modular RF Payload
- JSF ALIS
- proposal
- JSF/DQIM
- JSF
- CANES
- International co-operative research on process and methods for systems engineering
- Interface Stimulator Services Enterprise

Question 25: What was the Survey Responders primary role relative to the project(s)?

Results

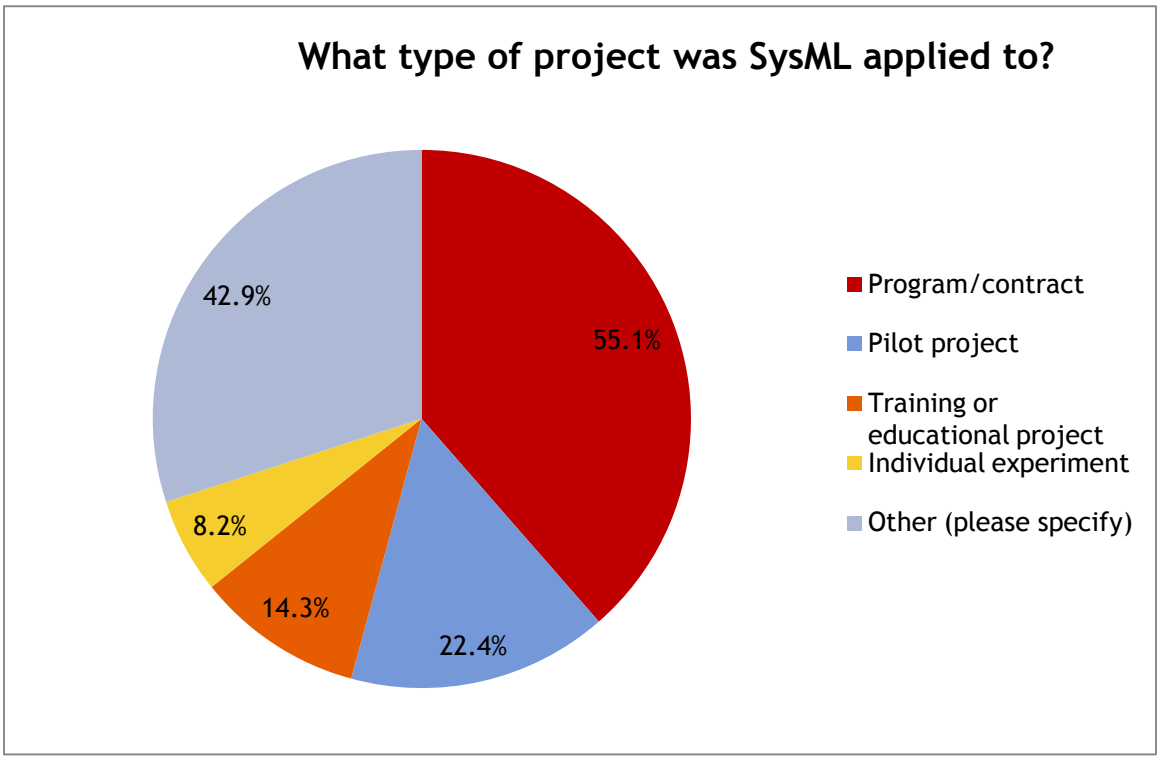


Open Ended Responses (Other)

- Supervisor of a Msc student, who was the modeler
- Going out the door
- I try to discover the best practices on "how do I do this?'
- Avionic Engineering research leader for Airbus & UML/SysML/MARTE Focal Point for EADS
- Pilot champion
- Teacher
- Mixed - I lead some and model some
- Methodologist/Mentor
- Model developer
- Requirements and design documenter
- I have also done modeling on these efforts and also consulting, so a mix.
- Software architect building a product line architecture
- Part of the Systems Architecture team
- Model Architect, Methodologist.
- Modeler also
- N/A Training
- Lead Architect

Question 26: What type of project was SysML applied to?

Results



Open Ended Responses

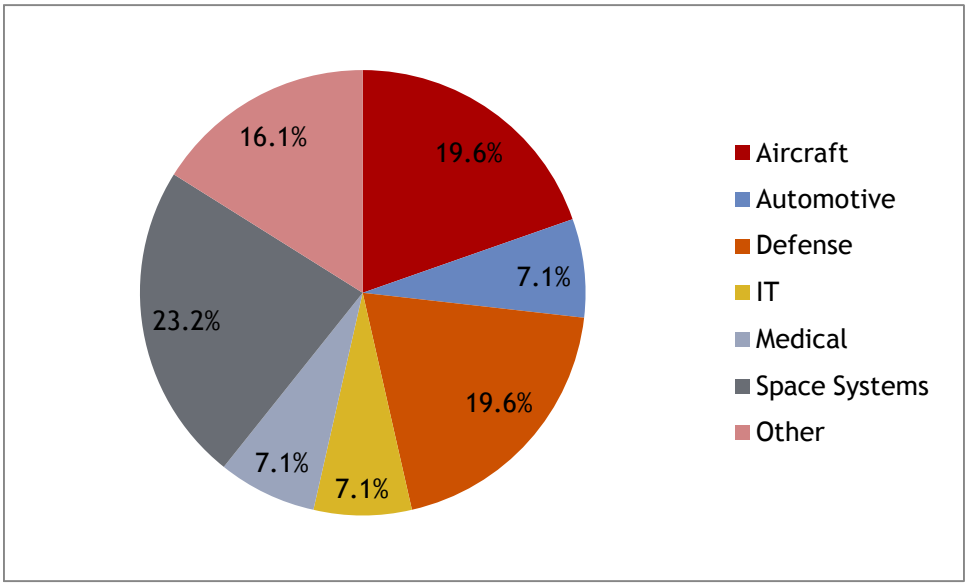
- Inhouse development
- All the above.
- There are several pilot projects - biomedical, biophyisics domain
- Msc thesis
- Previous Employer where they dared go forward
- Two projects: M&S Interface Standards; Real-Time, weapon Hardware-in-the-Loop simulation system rearchitecture design
- I have worked on various projects ranging from Training, Pilot Programs and Programs
- Pilot project executing within context of a real contract
- We are tool vendors
- All above
- Only R&D activities – no actual real project use yet.
- pilot, INCOSE Challenge Team, and JPL multi-mission project
- MBSE Challenge Team
- software development, medium-sized project
- All the above.
- This is based on experience using UML for systems engineering
- pre award phase of a program

- IRAD
- IRAD
- IRAD
- IRAD
- Several including educational
- N/A Training
- Test support tool being developed outside of standard program practices
- Pilot, Training and program

Question 27: What type of system was SysML applied to (e.g., aircraft, IT, medical equipment)

Results

Graph is manually developed from the open ended responses



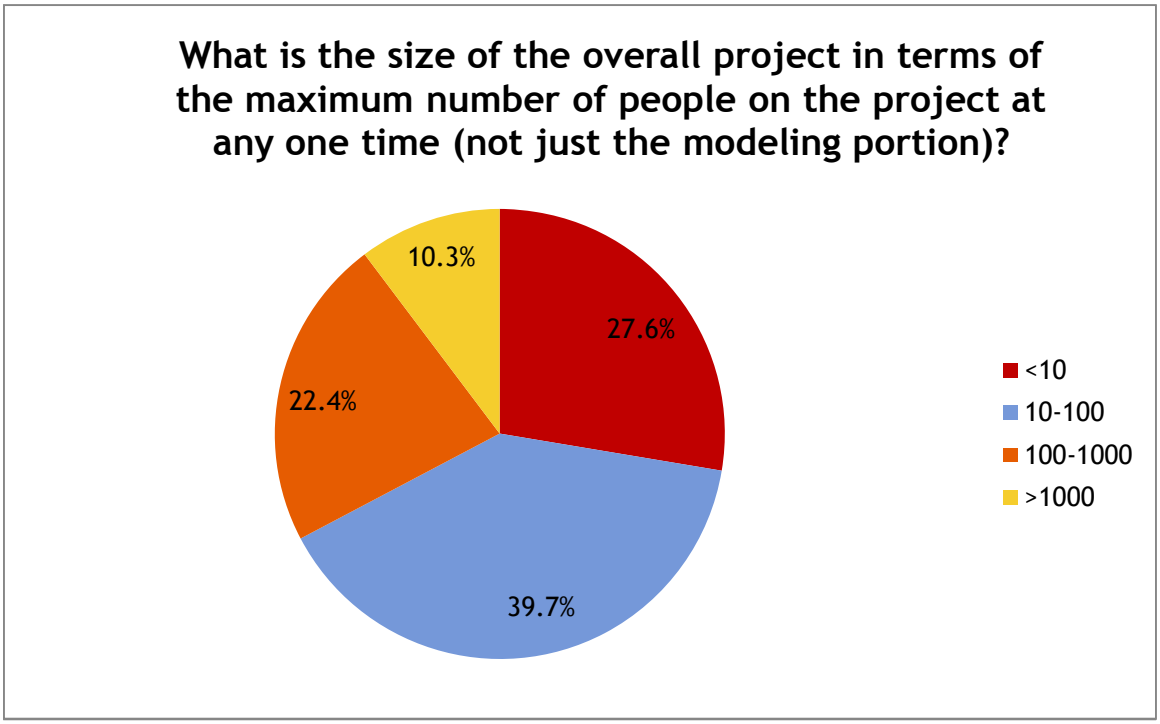
Open Ended Responses

- Deep space mission operations
- Medical Equipment, Data Warehouses, Enterprise and Process Development.
- Diagnostics, fMRI, MEG
- Aircraft avionics
- Meteorological system, automobile, clock radio, etc.
- Chemical system of a chemistry lab
- Mortar Fire Control
- Biomedical
- Aircraft avionics systems
- Weapon-torpedo
- aircrafts, space craft, missiles, satellites
- Automotive, rail
- Network design.
- IT, research and robotics
- Aircraft Test
- Radars
- Aircraft
- AEROSPACE

- Avionic computers & applications
- Automotive
- Remote sensing satellite ground segment
- Aircraft, IT, medical equipment, laboratory equipment, military equipment, telescope, mechanical engineering
- Space system development (as R&D pilot).
- Several and various
- SMAD Firesat text, Jupiter orbiter, deep space multi-mission operations
- Optical Telescope
- Aircraft and avionics development
- Telescope
- Spacecraft
- Involves XML, data transformation
- requirement-level systems engineering.
- Various: space systems, defense systems, water mgt. systems, automotive, excavators, mobile robots, green mfg, etc.
- Military simulation
- Aircraft
- Defense IT
- Logistics
- Manned Spacecraft
- Manned Spacecraft
- Satellite
- Aircraft avionics
- Logistics Data System
- Command & Control System
- Weapon System
- Weapon System
- Defense
- Aerospace
- Radar
- Aircraft
- Naval enterprise infrastructure.
- IT
- N/A Training
- Naval vessel
- Training systems
- Communications
- Network Management
- Space
- Command and Control

Question 28: What is the size of the overall project in terms of the maximum number of people on the project at any one time (not just the modeling portion)?

Results

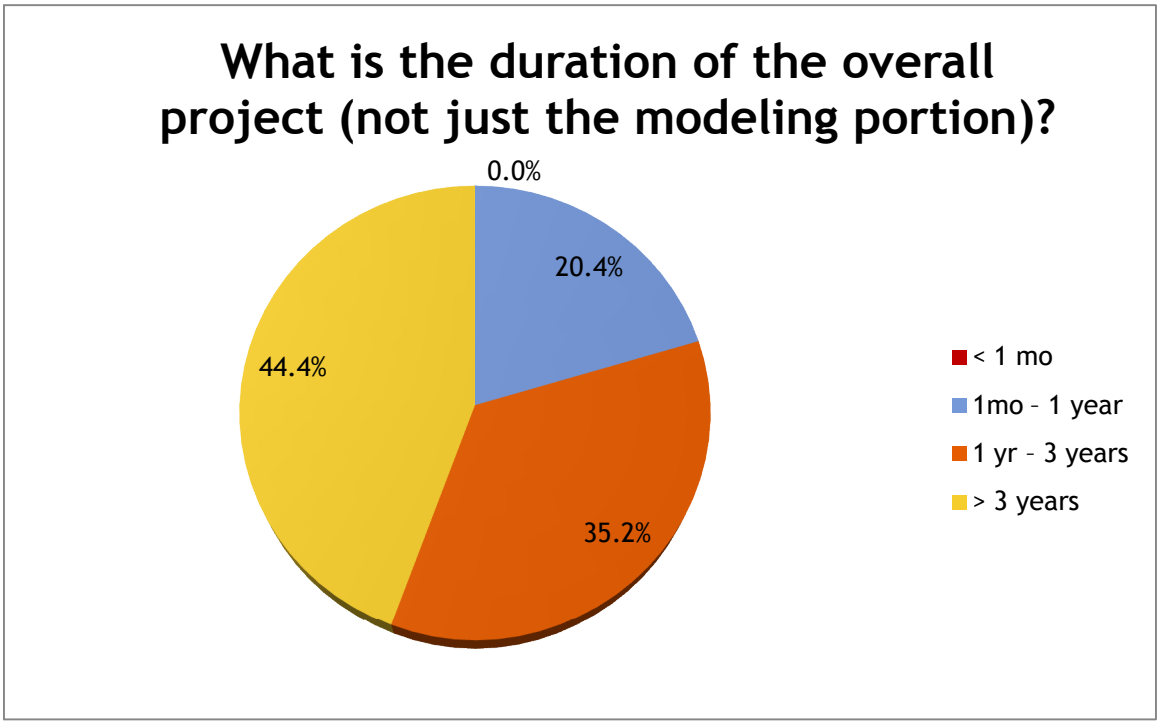


Open Ended Responses

- One project much larger
- > 10 and < 20
- Closer to ten.
- Size is limited for pilot project and will raise to 10-100 in the future
- Teaching classes and teams
- For all 3
- Varies - mostly first two levels; a few at third level (100-1000)
- Most of time, less than 10, but expands for customer review meetings.
- N/A Training

Question 29: What is the duration of the overall project (not just the modeling portion)?

Results

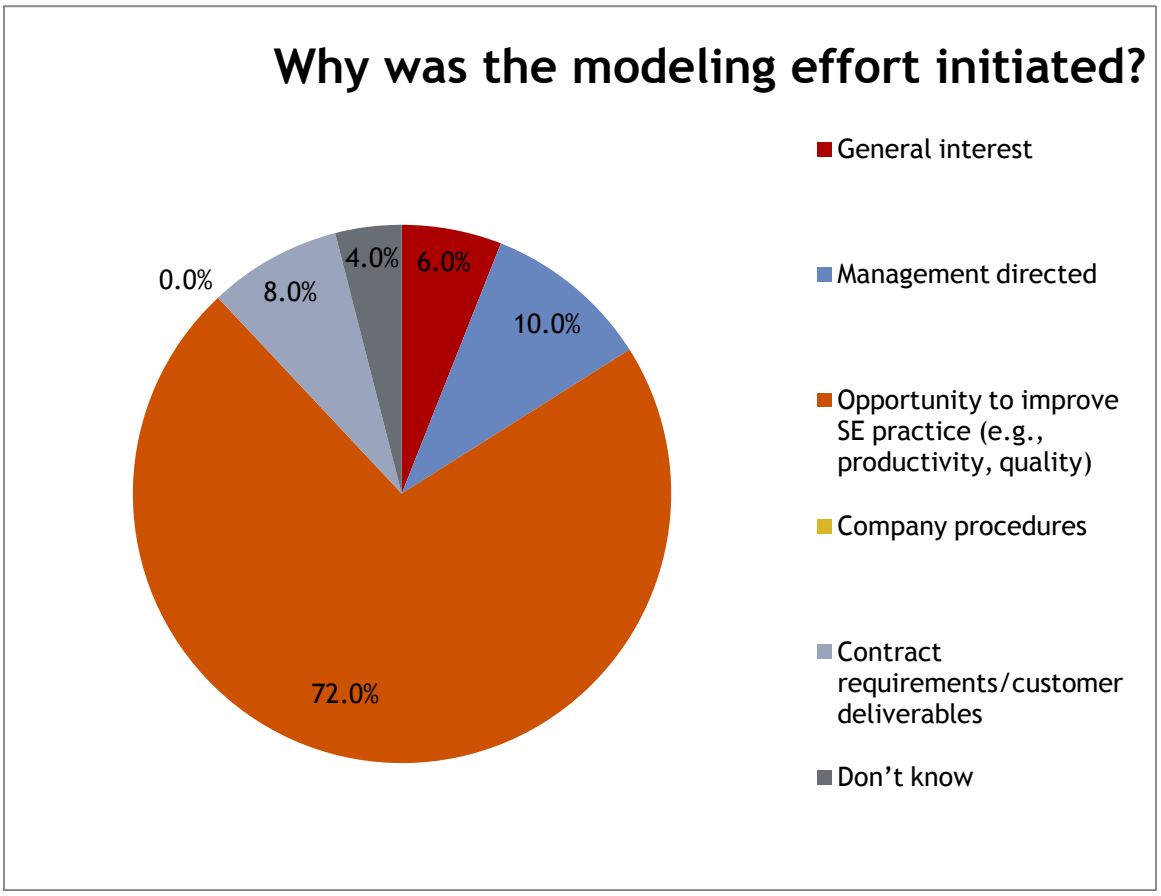


Open Ended Responses

- Again, this covers several projects
- Watts Humphreys worked with us some
- 3 months is an academic quarter.
- Most are very long, but I've done several that are about 1 year.
- If the maintenance is taken into account the duration can be more than 50 years (eg. 70 years for the A380 program)
- All above
- On going
- Ongoing
- Varies
- N/A Training

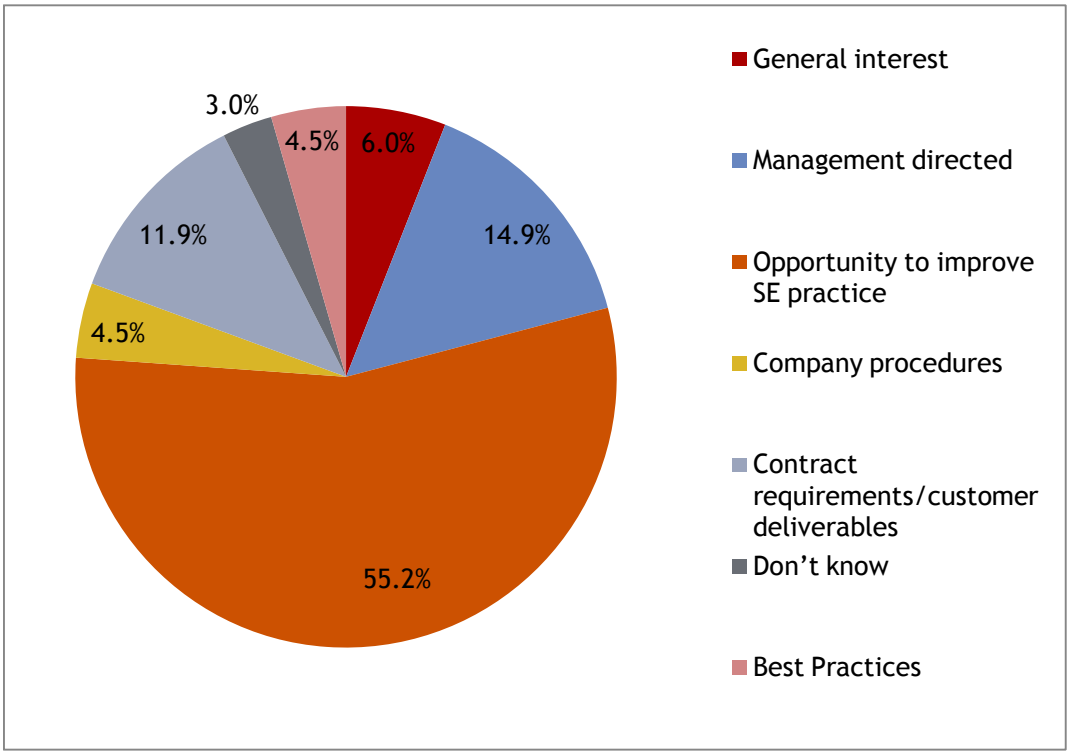
Question 30: Why was the modeling effort initiated?

Results



Open Ended Responses

Graph manually developed from open ended responses.

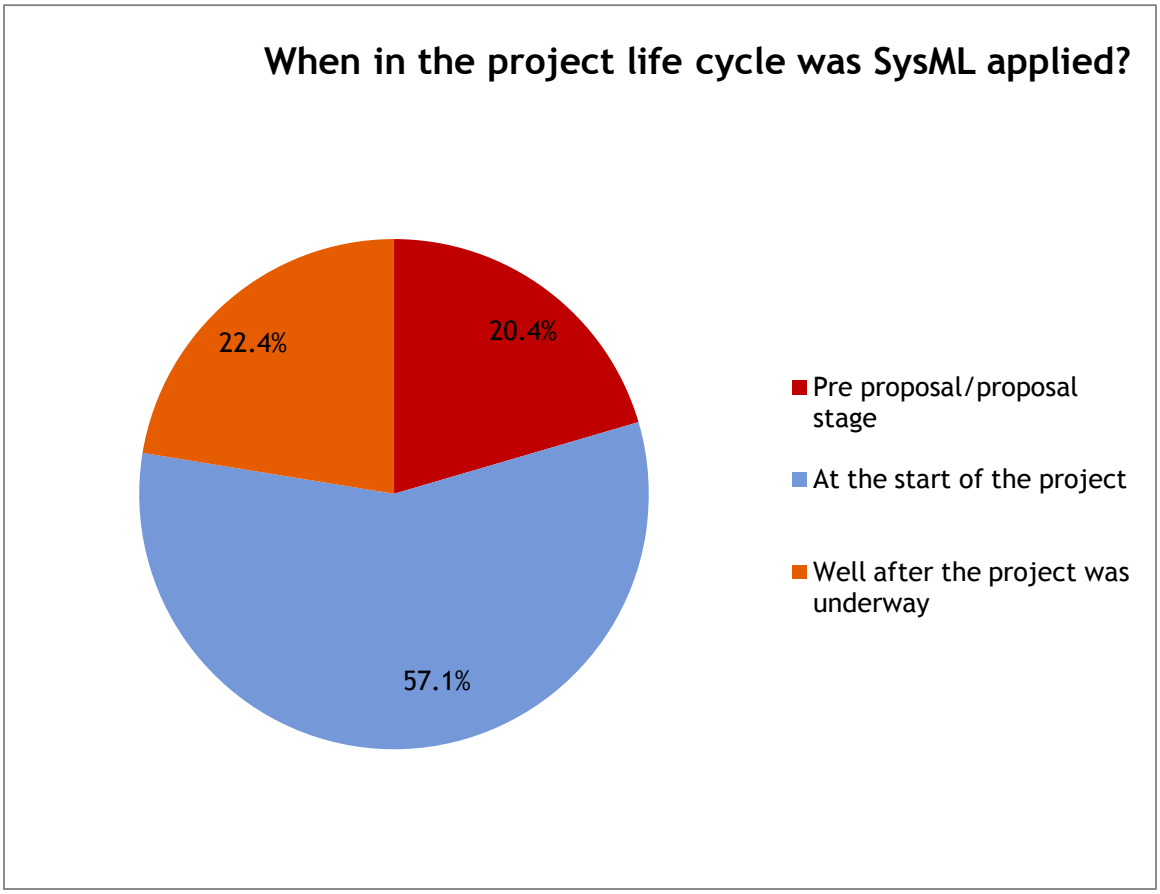


- Science drives management decision making
- Improve maintainability of requirements.
- Analyze adequacy of SysML and start SysML know-how transfer to our research group. Our group is from electronics and informatics.
- Experimental validation of models to data
- Looking for the best standards-based solution to develop standards and designs
- MSc. project
- The project team is looking for better techniques for architecting the system. Past practices are very weak in architecting.
- Functional Safety Management and "Total Traceability"
- All above (except Don't know)
- Research and Development
- MBSE teaching
- pursuant to documenting requirements
- Varies
- To bring down costs on a fixed price contract
- Management directed, Company procedures, Company procedures, Opportunity to improve SE practice
- I decided there was value in doing this.
- Opportunity to improve systems engineering practice; contract requirement

- Opportunity to improve systems engineering practice
- Both Management and Contractual requirements
- Customer directed
- Though I do think it may have been a management decision.
- And in some cases customer deliverable

Question 31: When in the project life cycle was SysML applied?

Results



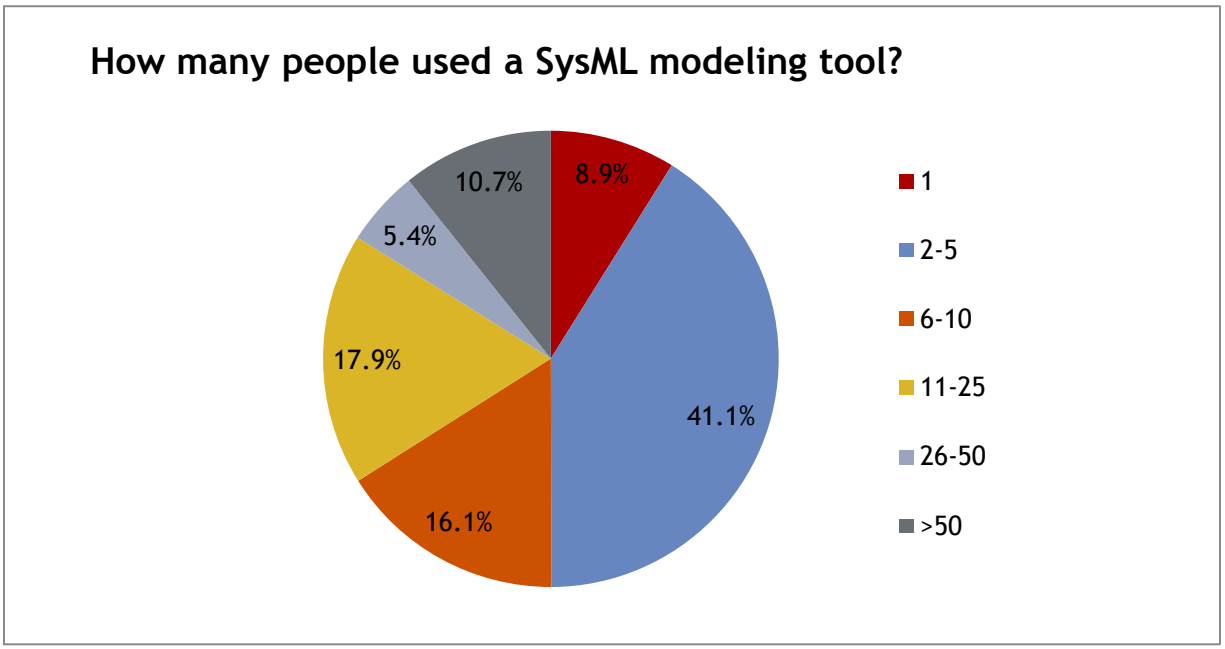
Open Ended Responses (Additional Comments)

- Not so much WELL after but certainly after.
- All the above
- MBSE is a core value
- Also as pilot for future projects.
- It was modeled an existent system
- Shortly after start
- All above
- ECSS Phase A (Feasibility) and start of Phase B (Preliminary Definition)
- All fialry early
- Nonetheless, a bunch of work was done in parallel before I could see it so perhaps "well after" is more true
- Varies

- SysML, although applied early in the program lifecycle, has not, yet been effectively integrated into the system design products as of PDR.
- N/A Training
- Various
- We started at the beginning of a new spiral, which was the fourth on the project

Question 32: How many people used a SysML modeling tool?

Results

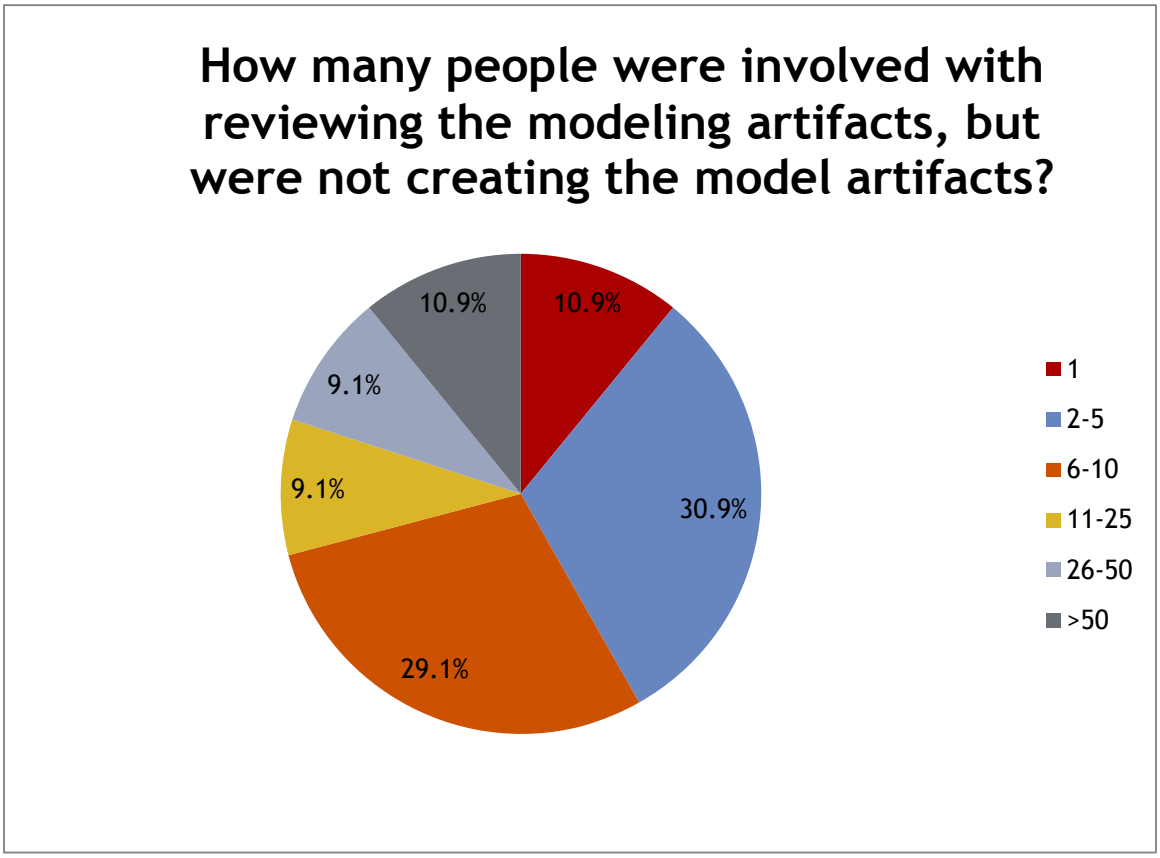


Open Ended Responses (Additional comments)

- Dragged SW team in for one team
- None. We used the definitions of SysML products.
- Total for the 2 programs, actually ultimately one designated user evolved for each of the two projects
- Limited number for pilot projects will raise to 6-10 per project and may be more in the future
- Many use a read only model viewer
- Most projects 6-10; some 11-25
- We actually used a UML tool but are considering using SysML in the future.
- Getting acceptance of SysML diagrams would be an achievement; not even trying to get others to use tools
- Varies - mostly in second, third, fourth levels listed above
- Rational Rose used for Use Case analysis and modeling
- Our user base is growing from 5 to 20 currently.
- Company wide >50

Question 33: How many people were involved with reviewing the modeling artifacts, but were not creating the model artifacts?

Results

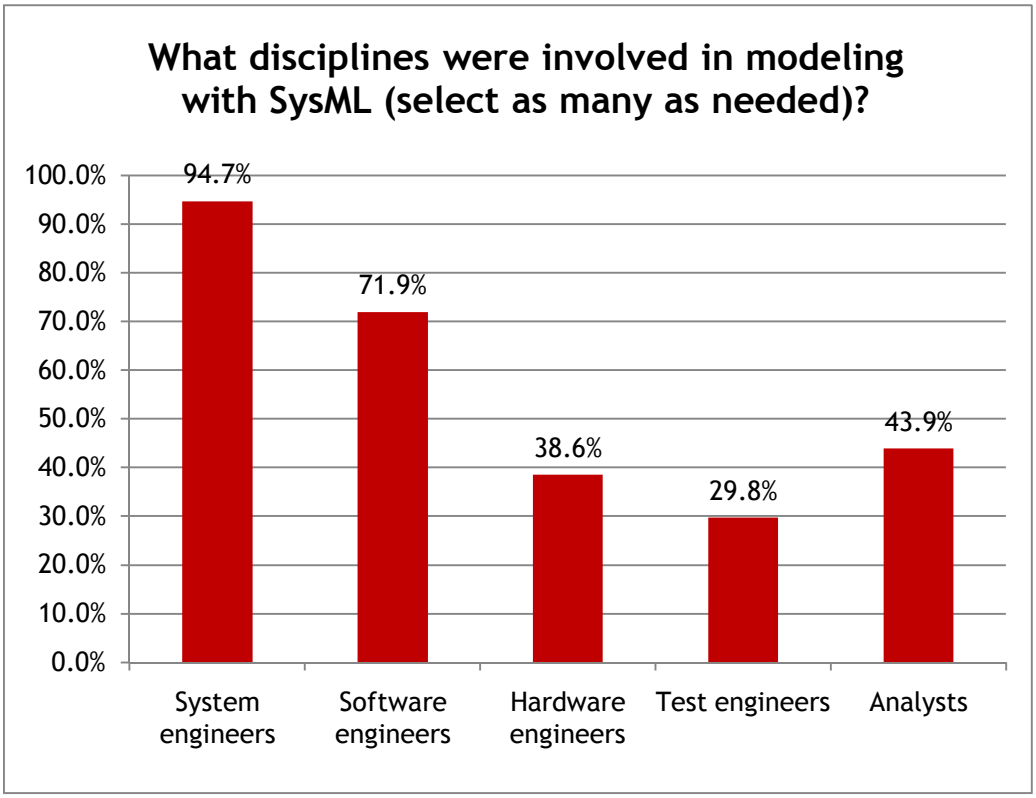


Open Ended Responses (Additional Comments)

- I didn't put this before - but SysML also needs some reviewing and approving notation. it would be cool of people could review and approve with notes etc and not disturb the diagram.
- Peer Reviewed everything with Cross Functional Team
- * Best guess
- Torpedo i/f standards ~15; HWIL Sim Re-architecture ~30
- Do not know
- Will grow in the future
- Most projects 2-25
- Model presentations don't qualify as reviews; reviews involve asking questions and checking somehow what the model does with these questions. In that context, there are few question-driven reviews currently; primarily because support for specifying reviewing questions varies amongst tools instead of being specification-driven (e.g., with QVT).
- Similar to #32
- Unknown as of yet

Question 34: What disciplines were involved in modeling with SysML (select as many as needed)?

Results

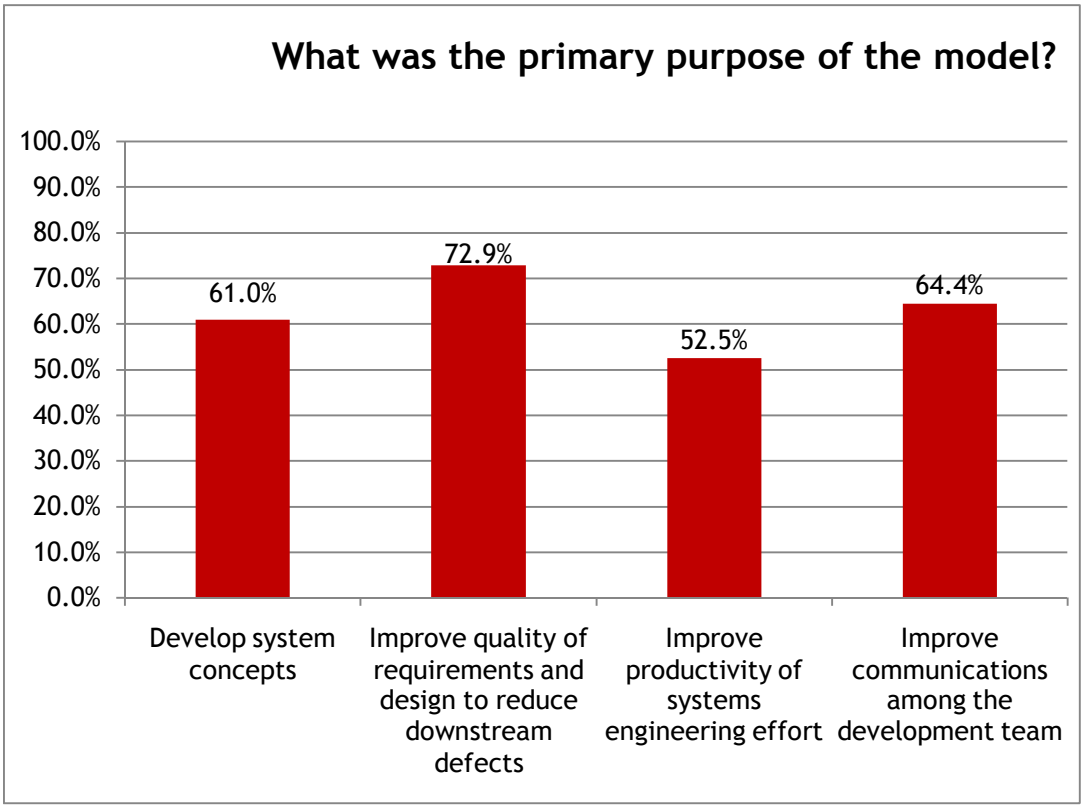


Open Ended Responses (Other (please specify))

- Doctors (MDs) Scientists (PhDs, DScs)
- Software as reviewer.
- Artillery SMEs and Customer
- Scientists and doctors
- Test is sorely missing
- Network designers
- Optometrist, Chemist, Biologist, Mechanical Engineers
- Masters/Ph.D. level operations research and IT students as well
- Architects, early phase mission specialists, aerospace engineers
- I am a tech writer but now that working with SysML, becoming a System Engineer I hope
- Varies
- Finance, managers, biz ops
- UI engineers
- Network and security engineers.

Question 35: What was the primary purpose of the model? Other (please specify)

Results



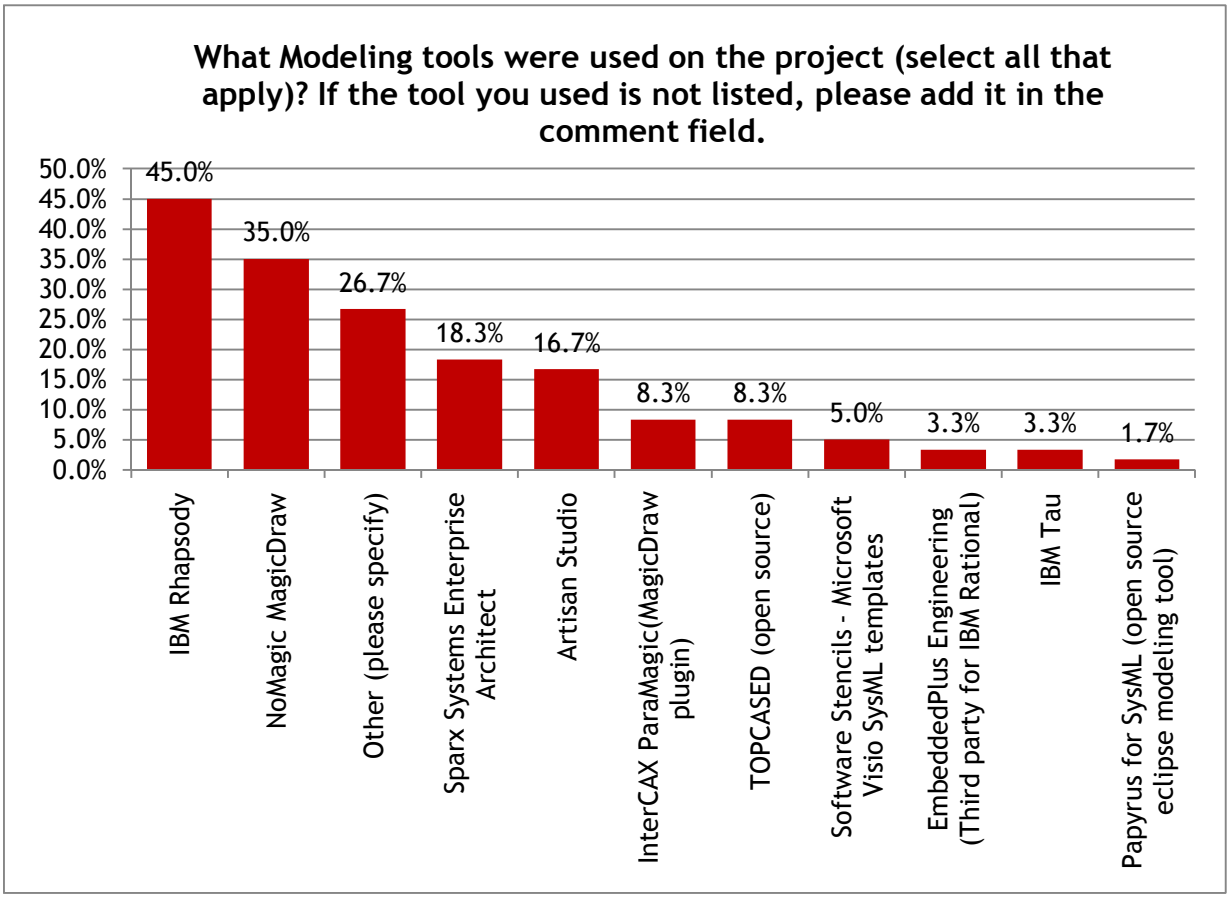
Open Ended Responses (Other (please specify))

- Provide a single definition of the entire system specification
- See 30. The chemical lab was also intererested in a model of its own lab to develop a remote experimentation system with the lab experiments.
- Did all four
- Group-development and capture of design
- Replace text based specifications
- Support network design.
- All the above!
- Development of system concepts is our long time goal
- In SSFRT to enhance the traceability between system and software
- path finder to model based engineering
- Develop MBSE concepts and best practices
- For architectural archeology in an effort to capture existing system architecture to proceed into developing a product line.
- To result in cost reduction by tweaking work flow process
- Documentation architecture. Develop design

- Reduce downstream defects
- Reduce downstream defects
- Reduce downstream defects;
- All the above

Question 36: What Modeling tools were used on the project (select all that apply)? If the tool you used is not listed, please add it in the comment field. Other (please specify)

Results



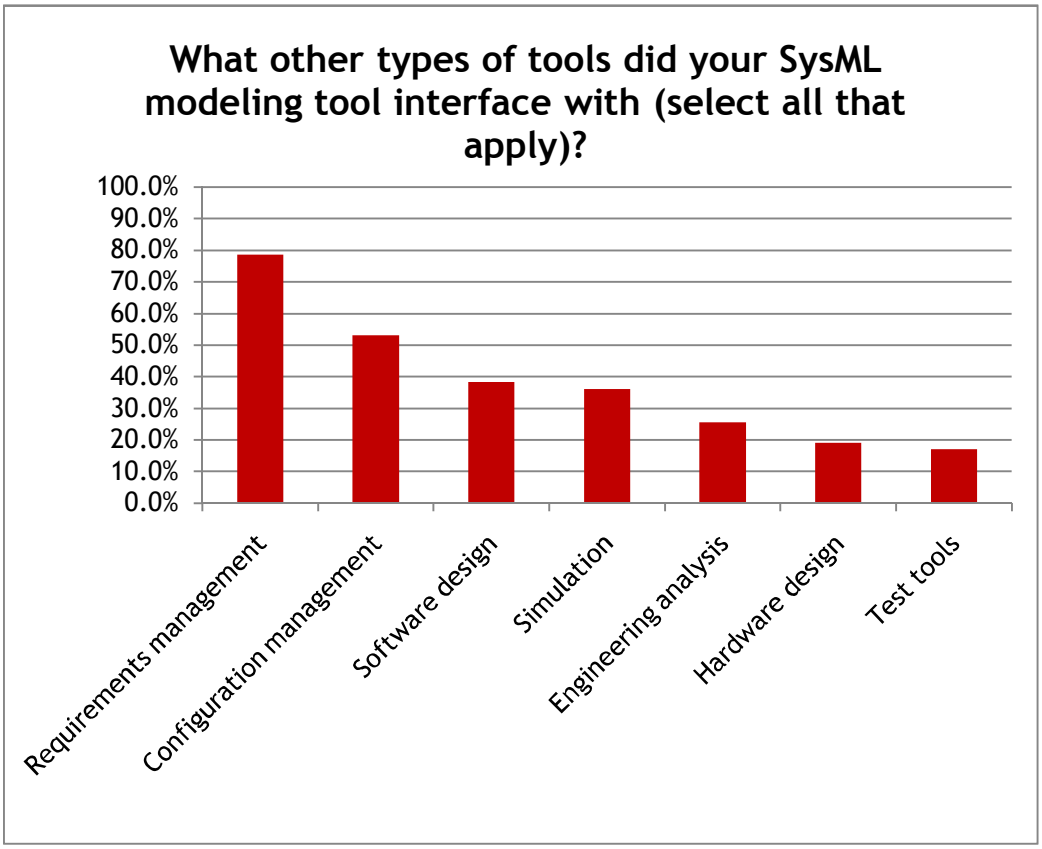
Open Ended Response (Other (please specify))

- WattWorks, AutoCAD 3D Mod
- IBM Rhapsody (Telelogic).
- Free
- Enterprise with transition to ArtiSan
- WattWorks
- With a little bit of IBM Rhapsody on the side, to prove some concepts
- In VSD proprietary R&D development on top of Eclipse EMF/GMF.
- Eclipse M2M's implementation of QVT; MOFLON.
- Rational Rose
- System Architece 11.2
- IBM (Popkin) System Architect

- N/A Training
- DOORS
- System Architect and DOORS and Opnet
- We used visio a little

Question 37: What other types of tools did your SysML modeling tool interface with (select all that apply)? Other (please specify)

Results



Open Ended Responses (Other (please specify))

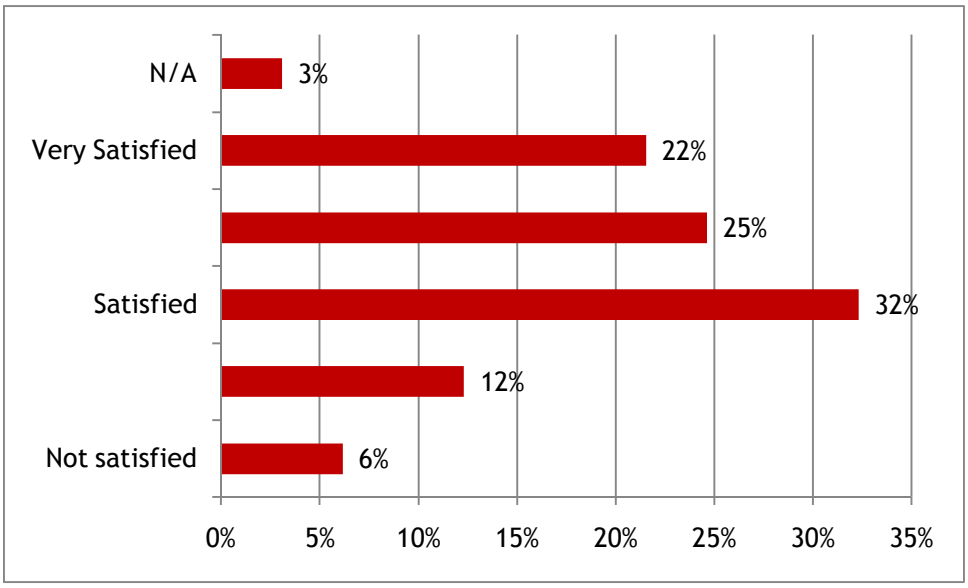
- Doors and Subversion.
- Hardware design was configuration evolution
- CM used, but not integrated with tool
- Attempted DOORS integration, but had major difficulties. Used Subversion for CM, which by itself creates some very interesting behaviors. Managing CM at the package level is a headache - with people tripping on one another to use an object in the library, but someone else has it checked out (but not really modifying that object!)
- ICD
- TOPCASED platform is evolving more tools and interfaces will be available in the future
- FMEA system analysis
- Very limited integration.
- Decision analysis tools
- Active decision to minimize the number of interfaces.
- We wanted to export created XMI files to some other tools but we didn't.

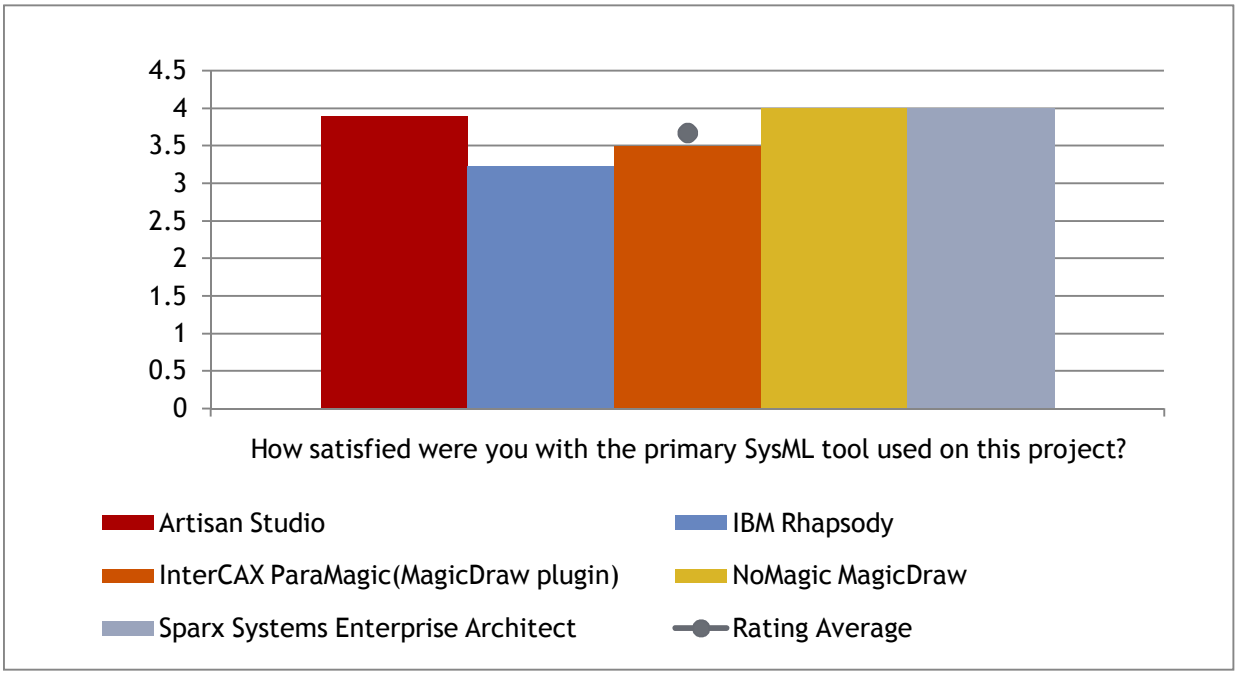
- Reporting, via tool-specific engines (e.g., MagicDraw's Velocity) instead of via specification-based QVT.
- Varies
- Interface Definition
- SW analysis also performed in Rhapsody in SysML, SW design in UML
- N/A Training
- Matlab

Question 38: Please rate the following: How satisfied were you with the primary SysML tool used on this project?

Results (Overall Average = 3.44)

Where 1= Not Satisfied, 3= Satisfied, and 5=Very Satisfied





Open Ended Responses (Please elaborate on your answer.)

- Paramagic is great. magic draw is very challenging using collaboration.
- Not satisfied by model checker performance.
- No simulation is allowed
- Had problem capturing all requirements from Sequence diagrams to Software requirements. Needed manual intervention
- Tool worked well once we established a good workflow. Before that, the team was flailing.
- MagicDraw will be the first choice.
- Easy to use, best customer support, best license manager.
- It's OK, but has many quirks that need to be addressed. Overall, we found this tool to be extremely flexible - but maybe too flexible, in some cases
- Despite the ergonomics and some diagrams that could be improved we have been able to model all or system specification
- Most projects did a good tool evaluation before purchasing the SysML tool
- Tools performed reasonable for R&D purposes, not good enough for production use yet.
- Magic draw would be a great tool if it didn't occasionally corrupt the model. this seems to eb specific to SysML models.
- Everything was addressed very well. But as the models gets very big the memory requirements increase dramatically (Java implementation).
- Lots of things to fix prior to full SysML compliance can be claimed.
- We actually used a UML tool but we are not sure if SysML can solve the problem.
- EA has a lot of merit, but it's not mature as a SysML tool -- maybe no tool is yet
- In late 2008 and throughout 2009, NoMagic has aggressively fixed a gazillion bugs JPL reported and produced no less than 4 significant updates of their SysML implementation & modeling tool. This is quite amazing in terms of customer support.
- Microsoft Visio is a generic diagramming tool; it wouldn't generate code for you, however, it's pretty intuitive and user-friendly. The SysML templates from Software Stencils were great, however a bit outdated or lacking the new ones from the latest revision.
- The goals of using the tools were not clear
- Artisan compliance with SysML was very good, the user interface had some issues
- Maintenance of the model was difficult and inefficient, and did not contribute to distributed engineering.
- Model primarily used for concept development for integrated scenario across multiple systems and execution authorities Focused on understanding and relating enterprise
- Leveraged lessons learned on Project D for developing Project E in SysML, especially Rhapsody modeling norms/cookbook.
- Great tutorials and example projects
- At the time, it was not very good, but it has improved
- Rhapsody is a horrible tool all the engineers on the project called it Crapsody
- Rhapsody 7.3 does not closely follow the standard.
- Use was superficial for documentation only.
- Rhapsody (7.5) is OK but it could be better.
- The tool was adequate, since it was the V1.0 of SysML release

- Lack of portability of models was a major deficiency. Obstacle to adoption and model re-use.

Question 39: What were your primary tool issues, if any?

Open Ended Responses

- Data integrity in collaboration repository. the tool corrupts the model frequently. it is borderline prohibitive to the adoption effort. also (and this is true for most tools)
- Some minor non-standard encoding methods in XMI
- Not satisfied by model checker performance.
- Consistency, ergonomy
- Architecture was not well modeled and ran loose
- XMI inconsistency
- The all-or-nothing aspect of the interconnectivity of names and types. Once I started to pursue the underlying model interconnectivity of UML/SysML, I would get errors and inability to do what I wanted graphically that I could not quickly resolve.
- The primary issue is that people would read "A Practical Guide to SysML" and try to implement their projects in Rhapsody. At the time, there were concepts that Rhapsody did not support in the same way that the book suggested. I had to help the teams change their workflow to make better use of the tool and avoid the limitations.
- Need more examples.
- Integration with other tools and many bugs yet
- None that involved SysML, only with DoDAF Profiles.
- When examining a requirement, there is NO means to navigate to the object that the requirement is expressing. Every other object has a "Links' page, except this one! Actors/ Blocks vs. Swimlanes (see prior SysML comments) Tool periodically crashes -no observable pattern yet. Several people experience this. Config control concepts at Package levels - what a headache. File centric tools like this are not conducive to large project teams working collaboratively. "Instance' vs. "Property" - had some peculiar problems with stereotype concepts. When a "block" was recast as something other than a block, the tool reverts to "Instance" creation, which caused some other rather peculiar behavior having "instance" objects in SysML and not "Property" objects (which is another oddity). Work around was to double stereotype a object as "block" and the recast typing. "Run Time Instantiation" vs. " Real Time Instantiation" - since most of these tools are based on software, instances of these things are assumed to be created for a run-time environment (where objects are created and destroyed on the fly), but we're trying to model a world with instances that just exist! Cannot create custom IBDs under a "property instance" to express a unique allocation - still trying to figure out how to express that the Left Mission Software Application is allocated to the Left Display Unit processor, which is allocated to the Left-side Display Unit (allocation of instances across layers) Cannot navigate from the Item Flow on an IBD to the intance property or block in the browser. Unable to show/edit Tagged Values in the Element List/ Tabular view - kinda defeats the usefulness of the tabular view! Very limited items available in Element List views For many items, when you create an object one way, the result is VERY different than if you create an item a different way! For example, when creating a flowport with drag and drop from the toolbar onto a property instance, the flow port has tagged values for direction. But when create a new flow prt from the Embedded Elements menus item, the tage values don't get assigned! Serious flaws with the information model behind the tool! Unable to resize a

Property on IBDs in one direction - always resizing in both directions! No means to change a object on a diagram from one type to another. Must delete the object and start over!

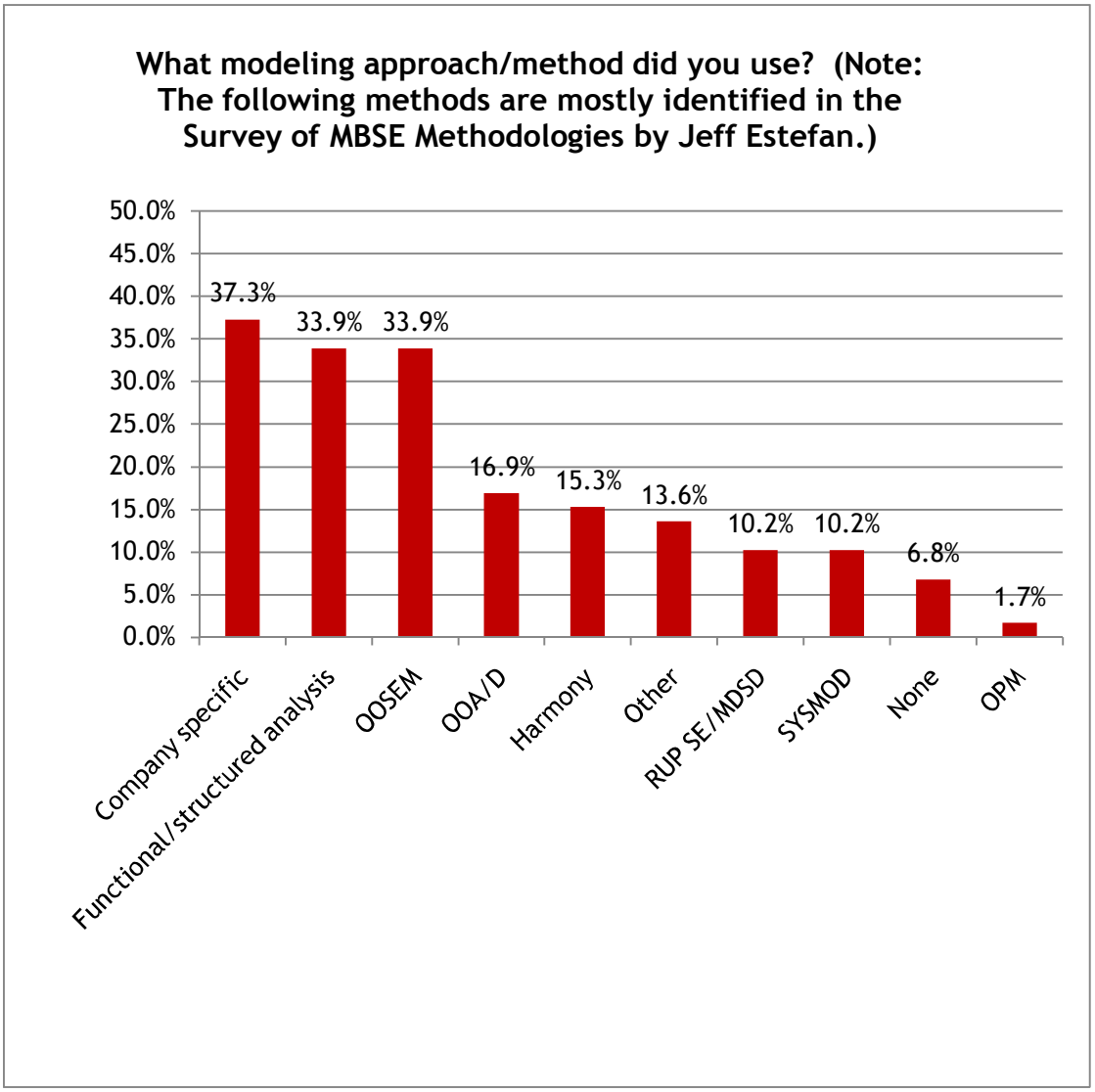
- Ergonomics
- Bad version control, baselining, configuration management and version compare
- Some incomplete support for language constructs (mainly with sequence diagrams). Some usability features (e.g. drag-n-drop available in some cases but not others). Some bugs Some tool performance issues (periodically hangs, periodically borks).
- It is difficult to generate useful reports from the model. - It is difficult to integrate other models, e.g. requirement models.
- TOPCASED: Insufficient stability.
- Use Cases
- MagicDraw - Interpretation of parts of the standard which are not precise enough, e.g. context specific values - Performance problems when model gets bigger - Configuration control difficult without Team Work Server
- Too few syntax checks when editing models. No support for allocating actions to blocks/parts
- Version Control could be better: identification of modified/locked elements, partial revert not possible, unaffordable model differing functionality. Navigation restrictions (missing bookmarks), diagrams size unmanageable. High complexity in configuring extraction of model elements to a document.
- It was not easy to define domain-specific models that can be interpreted and manipulated by the tool.
- --in the maze of menus, anybody's guess where to find any given feature --some things in the spec or in A Practical Guide to SysML just can't be done in EA --Practical Guide ought to be their bible, and they ought to have tutorials for doing everything in that book EASILY in EA ... instead, they seem to think it's natural that whatever you try to do is liable to be broken or at least require you to go to Tech Support

- 1) OCL This led me to collaborate with others writing a paper about requirements for an IDE for OCL, see: http://squam.info/?p=325 2) specification-grade metamodels & profile development. NoMagic's MagicDraw provides the best support for diagrams of all of the tools I've ever used. The open-source Eclipse toolkit provides the best support for OCL and QVT Operations of all of the tools I've ever used, including IBM's own Rational Software Architect/Modeler. 3) scripting. RSX supports scripts in Java as "pluglets" MD supports scripts in several languages including Jython, BeanShell, etc.. What we need, is support for scripts written in a specification language, i.e., QVT, where the integration between the scripting language and the modeling language is part of that specification rather than a vendor-specific extension.
- Steep learning curve, difficult to understand documentation. Limited useful examples to learn from.
- Finding out where to get for example, the {control} labels for pins.
- - Incomplete implementation of the spec. - User interface quirks. - Difficulty merging work from multiple individuals working in separate models (an admittedly challenging problem).
- Rose crashes often Rose is only UML based (no SysML)
- SysML standard not yet fully implemented. Simulation utilities do not fully support Flow Ports, user defined type libraries, and flow specifications.
- Already mentioned. I think Rhapsody is a great tool.

- The user interface could be slow and at times it did unexpected things, like not drawing a port on a block in the location I dropped it. That made it difficult to get the IBD to look as I wanted. Also a lack of a good report generation tool that did not require writing visual basic code is a major short coming. Another problem is that technical support people speak in complex software terminology and don't relate well to systems engineers, this inhibits communication and causes frustration.
- Incomplete implementation - more coming with version 11.4 next year.
- Diagrams easily disconnected from underlying model. Difficulty generating outputs.
- - Activity modeling in Rhapsody pre-7.5 is very tedious, requires many workarounds. - Clearcase very difficult to use in multi-site, distributed engineering environment... took 30 minutes to open model remotely!
- - Activity modeling deficiencies in Rhapsody pre-7.5 impacted productivity and constrained analysis.
- Would like more interaction with Simulink/Matlab Ability to read in Excel files. Document generator could be improved upon.
- still using some UML 1.x features, insufficient SysML capabilities
- Too was slow buggy, crashed a lot. Lost data on more then one occasion. A very poor user interface and bad workflow
- Activity Diagrams - some constructs missing (interruptible region, activity parameters, when an operation is used as an action number and type of pin are not enforced by the tool) IBD flow specs - not type checked
- No direct impact with tool use. Nothing was improved.
- Slow performance, too much clicking, complicated reporting tools.
- management of multiple linked models.
- The integration of the requirements tool with the modeling tool.
- Learning curve, lack of native support for the UML2 Testing profile stereotypes
- CM interface is a little cumbersome. Working collaboratively on the same project

Question 40: What modeling approach/method did you use? (Note: The following methods are mostly identified in the Survey of MBSE Methodologies by Jeff Estefan.) If you selected Other, please explain.

Results



Open Ended Responses (If you selected Other, please explain.)

- We used WattSys
- Used Structured to organize Use Cases which were modeled and reviewed as Sequence Diagrams
- WattSys

- We're trying to use a disciplined system engineering process that follows your typical "V"-model of development. Have run into problems with several vendor taught processes, like Harmony. Harmony assumes that you already have a design figured out before you've completed the analysis. Most processes are VERY weak at the up front analysis - giving Use Cases a cursory glance -or jumping to Sequence Diagrams which then adds your 'solution" components already before you've actually explored the usage scenario. We've found activity diagrams to be a rich resource, that most processes treat as 2nd class citizens or just brush over.
- In SSFRT: ISO 15288 and INCOSE System Engineering Handbook were merged with ECSS E-10 system engineering process
- system architecting methodology
- State Analysis
- Partially own SE2 method; Wymore MBSE
- slimmed-down version of OOSEM created for us by S. Friedenthal
- Our own method for integrating ontologies/metamodels/profiles.
- also ad-hoc methods (depending on the project)
- MSEM based, but tailored in order to accommodate an existing design.
- other published method
- MSEM
- N/A Training
- MSEM

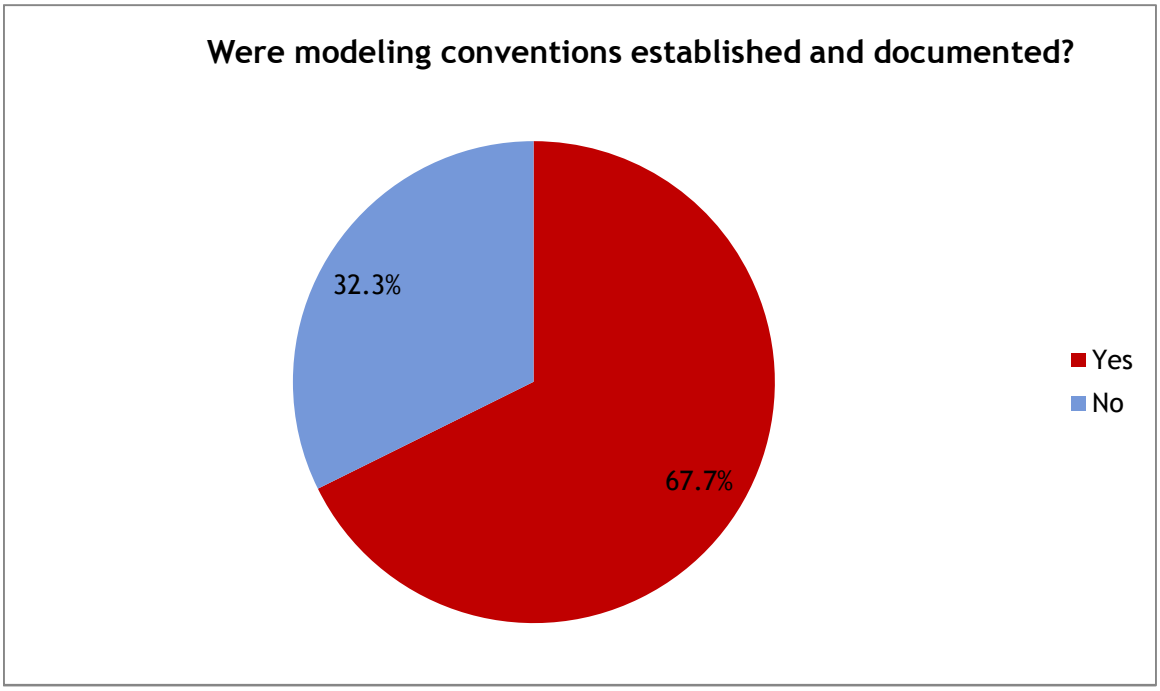
Question 41: If multiple MBSE methods were used, which was the primary method?

Open Ended Responses

- RUP
- Went for OOAD, but without quality control.
- OOA.
- in the end, a combination of RUP MDSD and Harmony
- SYSMOD
- Company specific
- Use Case to Activity Diagram (scenario) to Activity Diagram (data flow) to Software Architecture (application BDDs/IBD) and allocation from activity to Electrical view (processor BDD/IBD with allocation from software) to Mechanical view (blox BDD/IBD) with allocation from processor view - a complete, seamless transformation
- structured analysis
- SYSMOD
- N.A.
- SysML, UML, and IDEFx
- Company internal methodology
- It depends if we're talking about end-users vs. developers. For the latter, the driving criteria is to maintain parity of semantics/expressiveness of a given domain represented as an ontology, metamodel or profile extension of SysML. This is tricky to do but it saves a lot of surprises when questions arise.
- Other
- Functional Structured analysis
- Popkin ABM methodology.
- Functional analysis method, augmented by proprietary architectural techniques.
- Started to use OOSEM. For this particular task, it required some top down and a significant portion of bottom up approach. Need to find a good process for bridging gap
- MSEM
- OOSEM
- Company specific based on OOSEM

Question 42: Were modeling conventions established and documented?

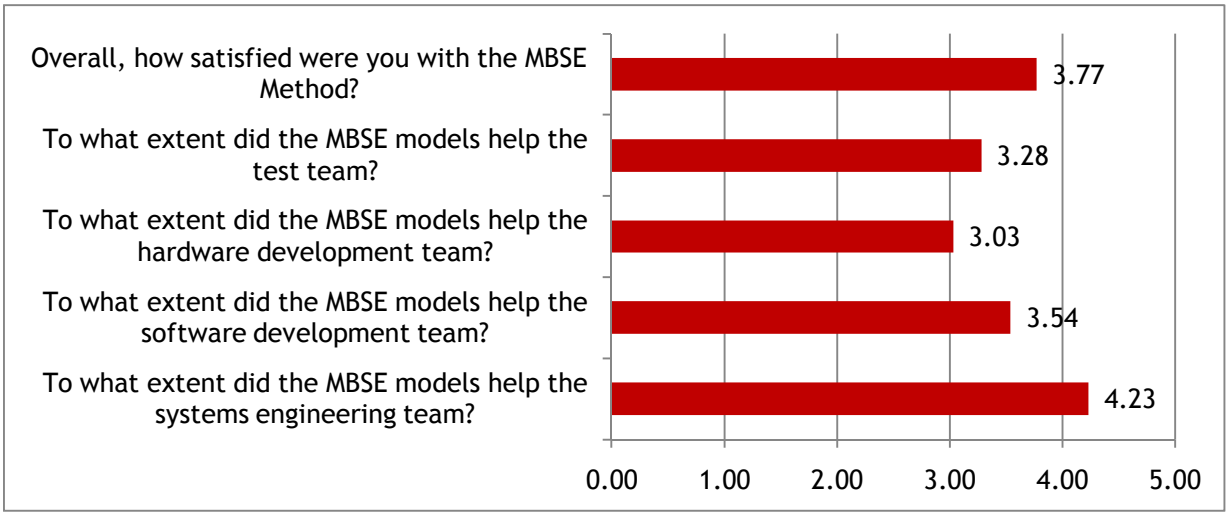
Results



Question 43 Please rate the following: (MBSE questions)

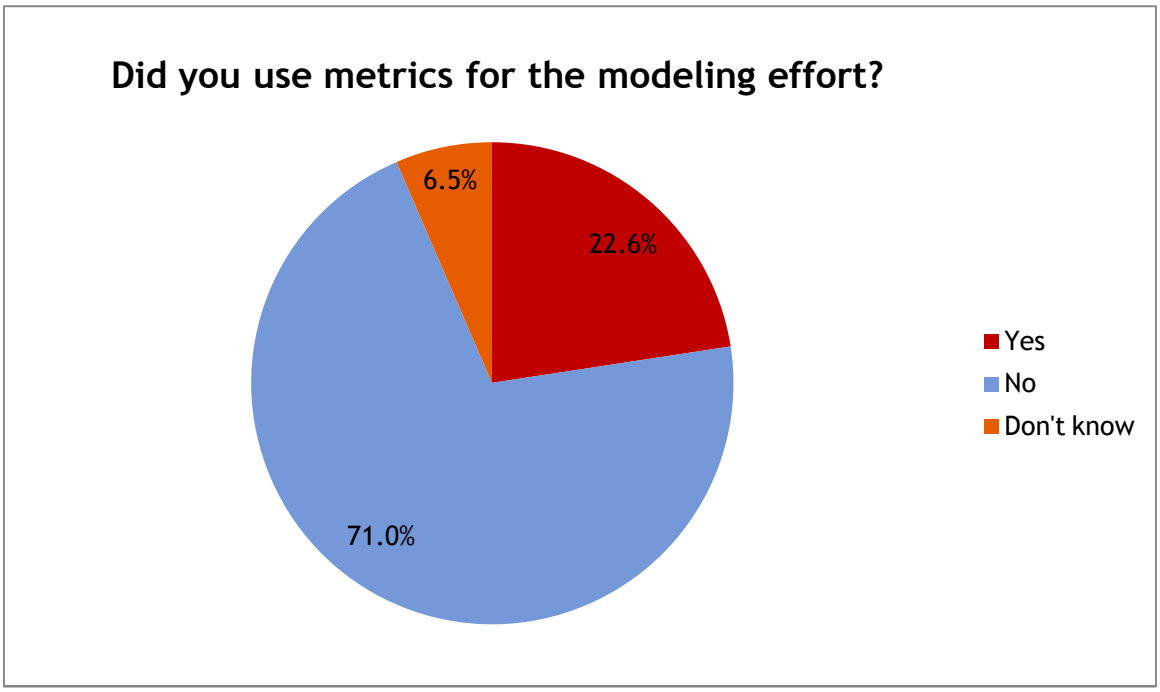
Results

Where 1=Not Helpful, 3=Helpful, and 5=Very Helpful



Question 44: Did you use metrics for the modeling effort?

Results

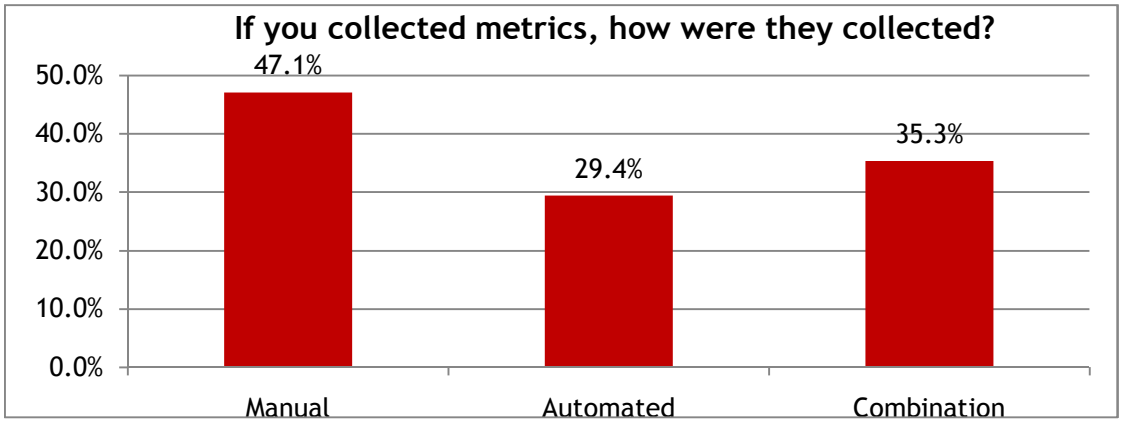


Question 45: If you used metrics, please list the metrics collected.

Open Ended Responses

- In our methodology, we develop (evolve) many models, and using cycles of forward and inverse Bayesian refinements on the models. The models are related to data and errors, then refined for probabilities such as likelihood and plausibility. Another metric was computational time on the hybrid, heterogeneous compute clusters.
- No - Because it was like Metric Police. They insisted on number of errors and time of reviewing - ridiculous
- Many, the most meaningful were likelihoods and probability profiles for outcomes, stability and kurtosis refinement
- Number of use cases Number of use cases completely described (activity diagram, sequence diagrams, and/or state charts) Number of blocks identified Number of blocks completed Requirements traced
- - Requirement satisfaction - modeling rule compliance
- N.A.
- - Time spent on modeling - Statistics on modeling elements, diagrams, etc.
- They are contract specific
- Number of use cases created.
- Model development metrics will be, but have not yet been collected.
- We were able it identify the number of IBDs and activity diagrams upfront, tracking to these numbers were a useful metric in determining progress.
- Hrs burnt
- Number of diagrams created at each layer
- What metrics are commonly used?
- By use case
- Use case development based on days - i.e., number of days to do a use case survey, number of days to write the use case, number of days to create the activity and sequence diagrams, number of days to complete the process by developing the organizational and behavioral diagrams to go with the use case effort.

Question 46: If you collected metrics, how were they collected? Please elaborate on your answer.

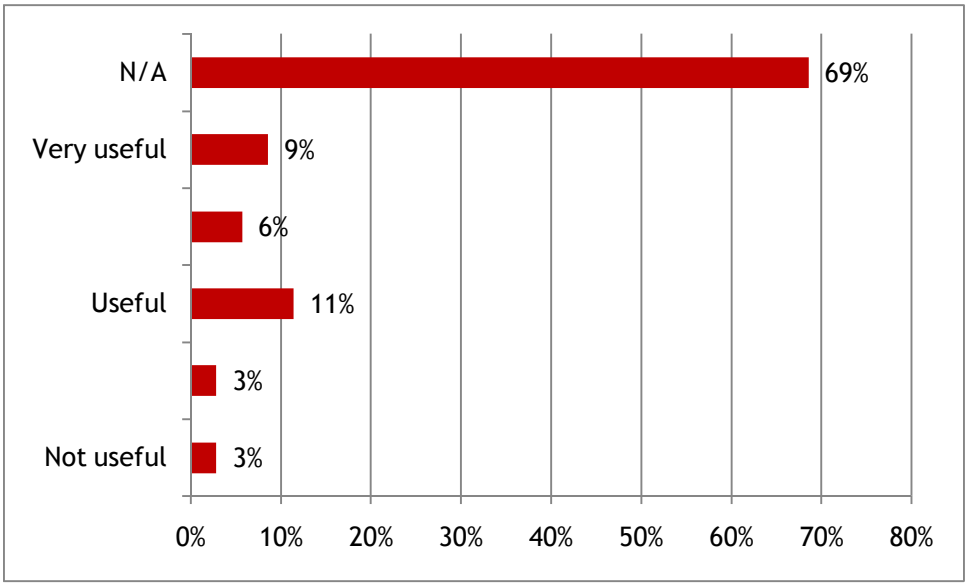


Open Ended Responses (Please elaborate on your answer.)

- Manually recorded into database
- Rhapsody provides an API that let us look at tag values. It was up to the team to set the tags correctly.
- We plan to use automated collected metrics. Works are in progress on that subject.
- A model checker, similar to a code checker (e.g. QAC) was realized in house based on our modeling rules
- N.A.
- There were requirements allocated to use cases but the metrix was not maintained.
- N/A Training

Question 47: If you were responsible for analyzing the metrics, how useful were they?

Results (Rating Average 3.45)



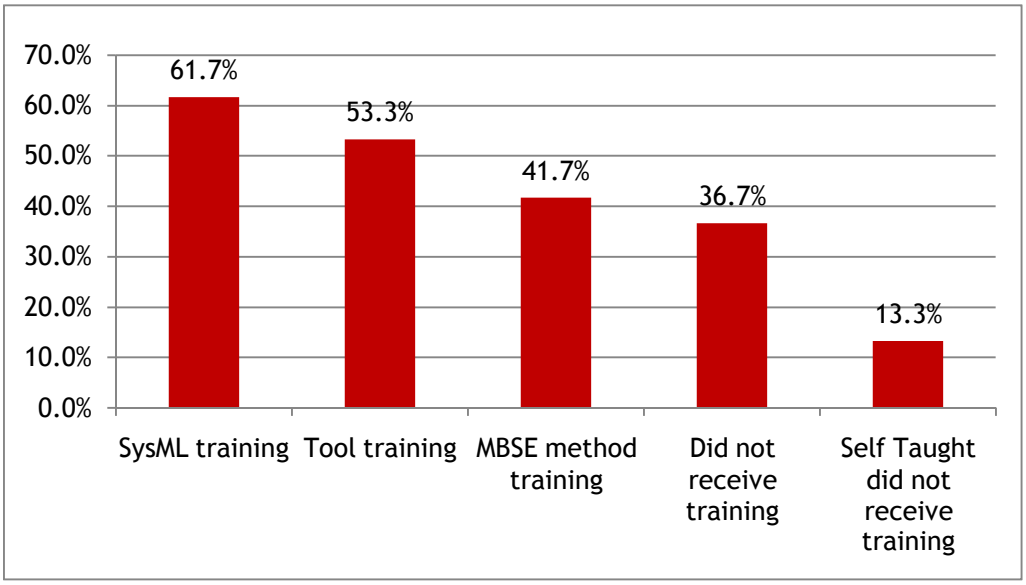
Question 48: What were the primary metrics issues?

Open Ended Responses

- Most members do not understand Bayesian metrics
- Inflexibility of metric analysis. Some SW Engineers threatened to insert errors to meet inflexible goals
- I would call this a CMMI Level 3 effort. I don't have enough experience to determine which of these metrics were statistically significant.
- N.A.
- Too vague and not continuously recorded
- The metric thresholds should be evolving over the contract terms and hence the values should not be high while customer value low
- Metrics where used to 'check the box'
- Team leadership did not sponsor metrics collection.
- No way to measure engineering productivity through SysML

Question 49: What type of training did you receive?

Results



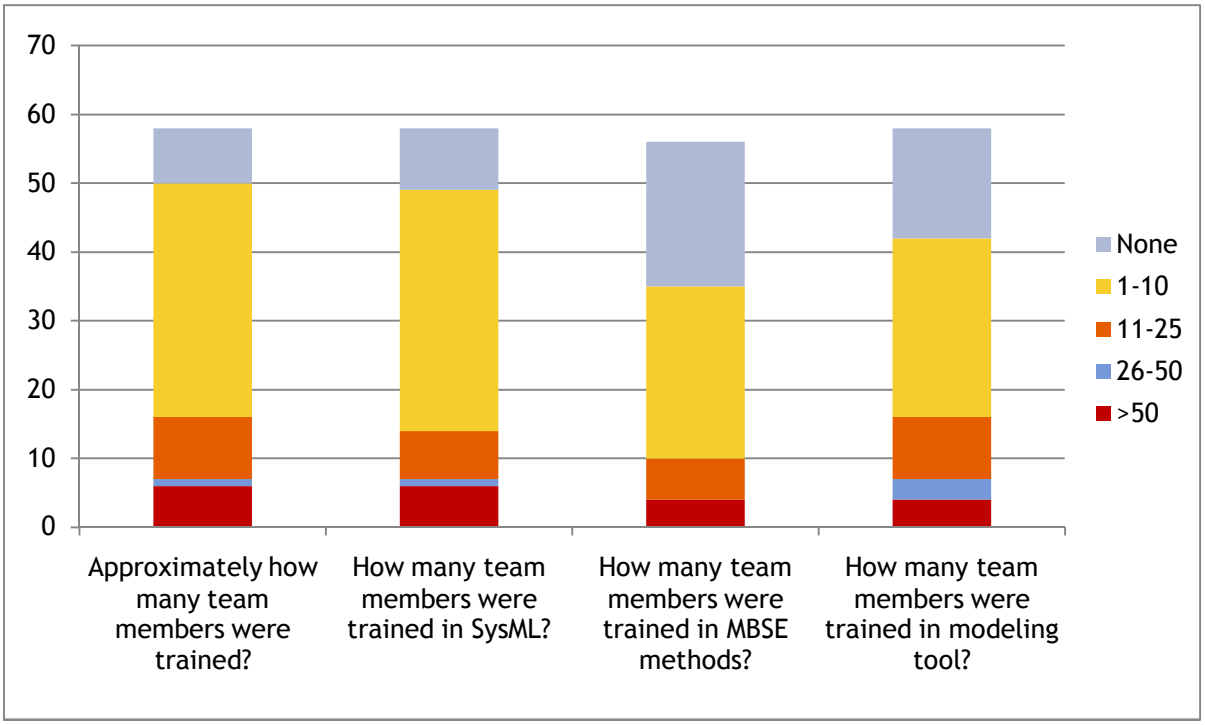
Open Ended Responses (What other training did you receive?)

- Bayesian inference.
- SysML for Rhapsody.
- Also had at least 2 one day sessions with Watts Hunphrey on SW Personal process and Team Process. also went through BP Douglas's Book on OO Design
- Training from implementing the UML/SysML specification in the tools and developing the mapping functors to the graph foundations.
- UML and software
- Enterprise Architecture Training (DoDAF)
- All the training came from using SysML in a training environment of trying to figure out best way to use the SysML tool and SysML for system development.
- Had vendor teach SysML, and tool training, as well as a methodology that he advocated - which does not work for us (It's just wrong from a system engineering perspective). So, we've adapting our RCI processes and learning how to apply SysML to that process.
- UML training, ten years ago.
- I have learned SysML by myself and made my PhD thesis about test case generation from SysML activity models. I train people in SysML and our MBSE methodology in a two-day training.
- Self trained team. Bought SysML textbook. Studied tutorials and published models. Attended SysML Info Days Dec 2008. Developed (and continue to do so) style and modeling guidelines to guide additional team members. Otherwise all "on-the-job" training.
- ESA staff self-trained through SysML tutorials from OMG SysML website and textbooks.
- I attended S. Friedenthal's conference and one private presentation, and a few others ... was the beneficiary of many answers and much mentoring from S.F.

- Trying to resolve issues in OMG specifications is a very humbling and enlightening experience.
- mostly self-taught.
- Read OMG Systems Modeling Language tutorial from INCOSE 2006 Conference; reading Sanford Friedenthal's Practical Guide to SysML.
- Systems Engineering Master's
- I had UML and SysML training prior to starting the program
- I trained the rest of the team in modeling basics/tool use. Mostly BBD, IBD, Interfaces and activity diagrams.
- Not everyone on the team received tool training, but those that did were able to support those that didn't
- Training was limited to what was in each tool's "help" or through "Google" searches.
- DoDAF, IDEF0, and requirements development. Also tool training.

Question 50: Approximately how many team members were trained? Please elaborate.

Results



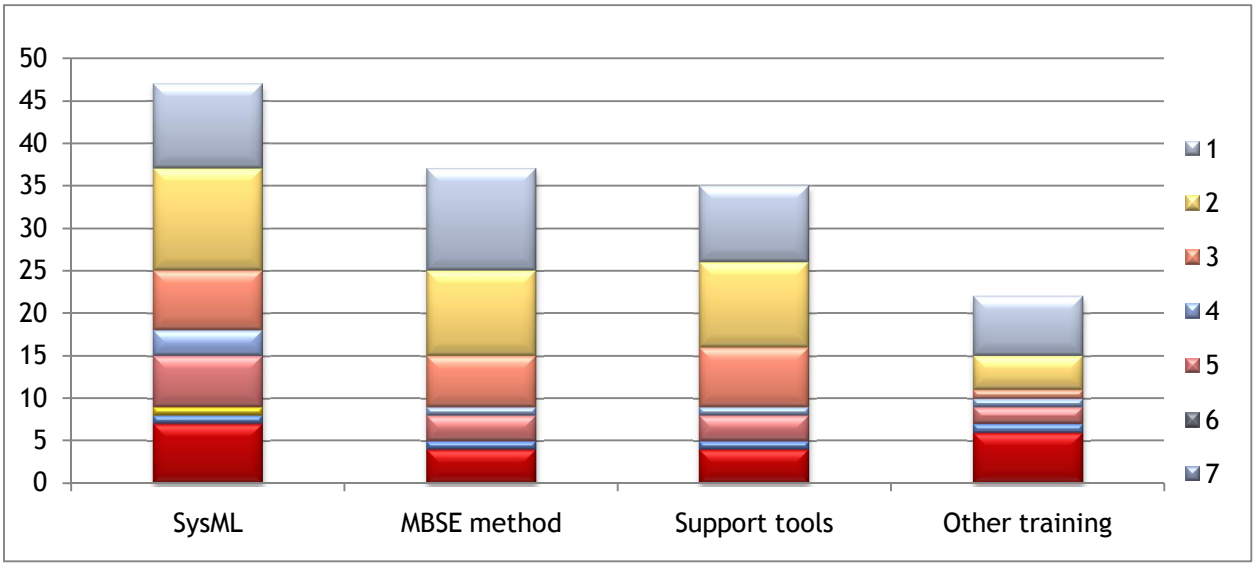
Open Ended Responses (Please Elaborate)

- Method training was not formal
- Everybody did the Team Process session. Only SW did Personal Process
- The customer did not want to have any workflow training because "we have our own processes." As a result, they floundered for a while.
- Self-Education: guides, manuals.
- We had 2 modelers for the whole group but gave quick training of MBSE OOSEM constructs and familiarization with tool.
- Our specific training includes all aspects: language, method and tool
- Same as answer to question 32.
- Limited with regard to. SysML, formal training was UML based.
- Undergraduate, Masters, and Ph.D. level students
- Training program is still being rolled out. Strong interest.
- We need a language/tool that does not require intensive training.
- As above, I am the sole SysML person in my organization ... also the sole tech writer
- JPL offered a lot of training on SysML and MD to many users.
- There was one engineer on the program that received training with ROSE. I had had training from other companies on other tools. There is in house training in UML and SysML but the training was not required.

- ABM did not use SysML
- Factory on-site JIT training, based on sample problem from relevant application domain. This was very successful!
- Leveraged INCOSE/OMG SysML training for the team. Didn't do the rigorous training we did previously on IFCEC.
- Everything was self taught. I would highly recommend “A practical guide to SysML”. Learned a fair amount through Rhapsody and Artisan Tutorials and discussions with Artisan app engineers.
- You should have a ~1 level. The training is very slim.
- Lead had SysML and MBSE training. Principle modeler had UML background and OJT on the tool and SysML
- I give the tool training in the context of SysML.

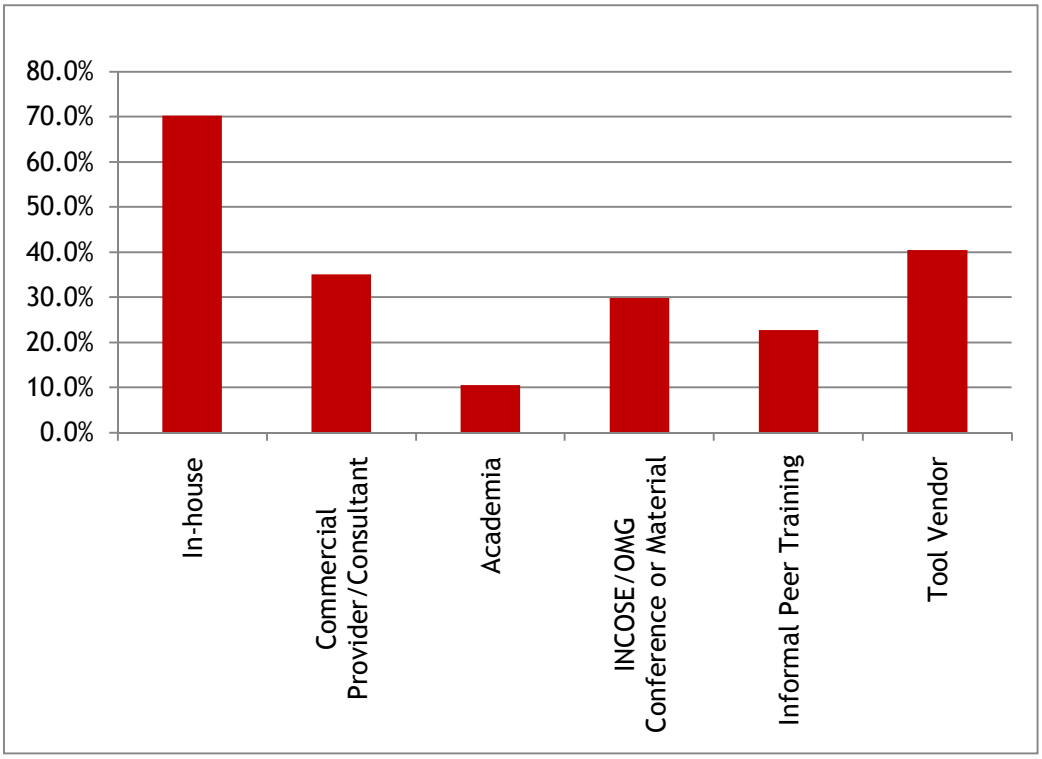
Question 51: How much training was offered in number of days ot the team members involved in the modeling effort?

Results



Question 52: Who developed and delivered the training? Other (please specify)

Results



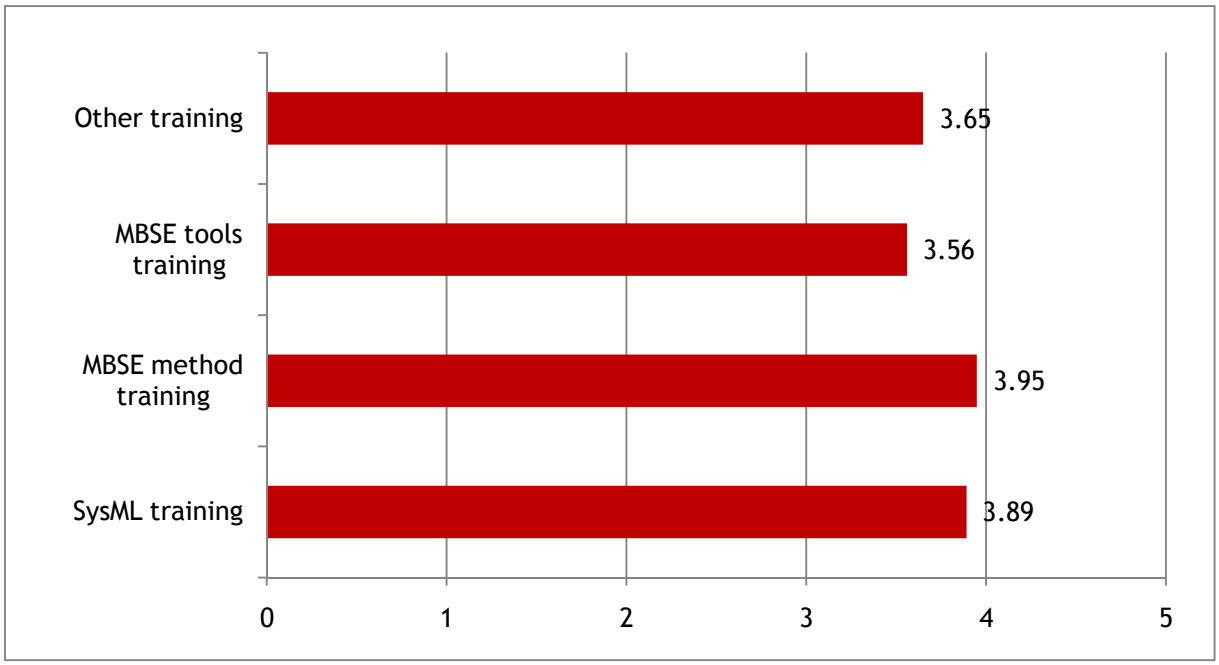
Open Ended Responses

- IBM
- Watts Humphrey, Steve Janiszewski
- Self-Education: guides, manuals.
- OOSE for SysML and MBSE / tool vendor for tool training
- In house methodology parts combined with standard training material
- We did not do training.
- MBSE is in-house training (JPL's State Analysis)
- We offer training
- Feabhas
- SysML training was provided by our own people
- Me

Question 53: Please rate the following: (other training, MBSE tools training, MBSE method training, and SysML training)

Results

Where 1=Not Useful, 3=Useful, and 4=Very Useful



Open Ended Responses (Please elaborate?)

- Course in OOD
- SysML Rhapsody training (IBM)
- Didn't have specific MBSE Tools.
- Maturity level of existing training programs was low.
- There was no other training.
- Did not methodology training.
- Until you start using the tools you really do not get a good grasp on the subject.
- Project team loved our in-house workshops!
- Training and support was the key to success for our pilot projects
- Answer is based on the training evaluation feedback from participants.
- Self-training through reading and tool trials.
- Difficult to evaluate what you have created yourself. Students appear to be productive.
- Returning to my pet theme: SysML needs tutorials in plain English; otherwise, most OMG presenters were superb
- I don't have results about the effectiveness of training from end-users.

Question 54: What were the primary training issues?

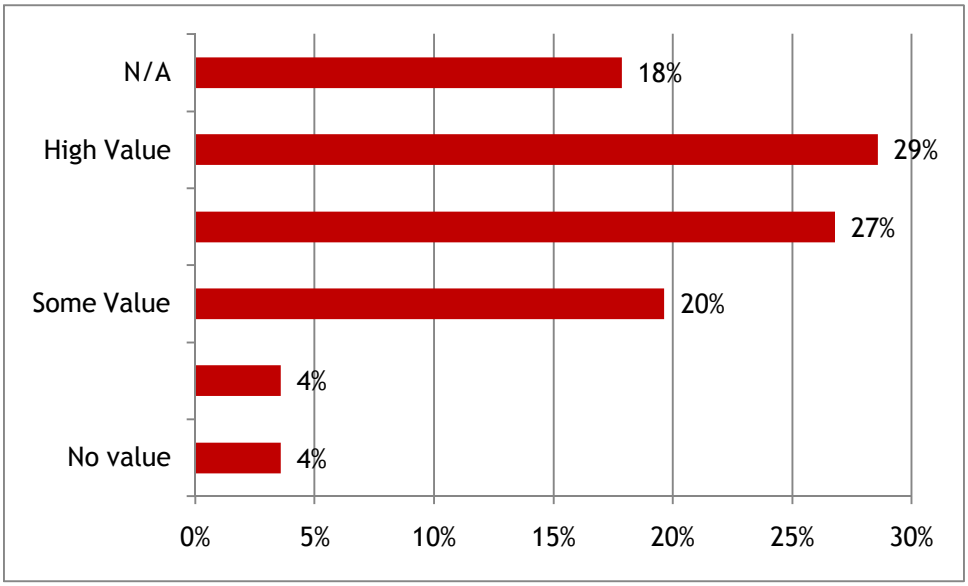
Open Ended Responses

- Commitment by Management
- The nature of the projects require "different" MBSE than say a toaster or a building. Human physiology and related biomedical systems are complex, adaptive systems.
- Tool familiarization.
- Systems was not trained in PSP.
- Alignment of training with a more scientific paradigm.
- There were no training issues.
- People not showing up for the course. The client thinking they knew what they needed. In house expert who knew UML and some SysML, but not the tool the group was using causing many problems.
- Not enough availability of SysML Training through INCOSE - One class offered by S. Friedenthal was immense value through INCOSE-LA Needs to be on line and via Pod Cast
- Not enough time to train the necessary people.
- Instructor knows a particluar methodology, but lacks practical experience in real-world engineering. Instructor was not the greatest teacher.
- OO aspects. People have to change their way of thinking.
- - Methodology - Introduction to SysML - Introduction in Enterprise Architect - Exercises
- Lack of local source of training, limited budget combined with high cost to go elsewhere for training was deterrent to getting formal training.
- It is hard to transfer the content of training to the projects without having any consulting.
- Lack of body of knowledge and good examples for space system domain.
- Consistency of grammatical approach in terms of teaching and application
- Finding a small but still relevant example.
- Difficult to support the 3 pillars at the same time: 1) Language specification 2) Practices and Guidelines, modeling concepts applied to a real system 3) Tool usage
- We want a tool that everybody can use by just playing with it (like Microsoft Word or Internet Explorer).
- I would prefer that OMG or someone establish a canonical set of simple exercises and teach them ... I guess the posted tutorial attempts to do that, but of course the posted version lacks audio, and in general the material needs to be packaged for effectiveness
- Methodology training is lacking is SysML. Since there are no normative criteria for analyzing SysML models in the specification, it isn't surprising that end-users feel wanting for some training on how to analyze SysML models. Little do they know that no such training is available because analyzing SysML models is, currently, a very tool-specific matter unfortunately.
- People without any OO thinking background have a steeper learning curve
- Difficult to absorb training if trying to learn SysML at the same time as tool training.
- Inappropriate scope for project engineers.
- Not enough budget and need CBT type of training
- Multiple sessions over a longer period of time are needed to allow students to digest the training in bite sized pieces. Training in the Language, the Tool, and the Process are all necessary and are not always provided together.

- Trainees are not ready to take tool training because of their lack of OO/UML/SyML bkground
- Retaining those people that were trained. In some cases in the time between training and using the tool the staff changed and we coudl not afford a second round of training.
- Training was over a year prior to application.
- Method tweeks needed to be applied, and resulted in the "Modeling Cookbook" of norms and conventions, which has been used as a basis for similar modeling efforts.
- Distributed team, experienced modelers. Training tailored to expertise of modelers, much hands-on & coordination.
- Not enuf time
- SysML offers no real help understanding any given domain, can't model what you don't understand regardless of how good or bad a notation is. SysML trainning is not really what the team needed.
- Constructing a class tailored to project
- No integration. was presented in wrong order. no central expert to advise on order of training.
- Teaching the new systems engineers how to produce good designs based on syntax, semantics, and aesthetics.
- Time and money
- Money and time, we should have budgeted more for training. Then it is hard to get the team to a training class for a week, so we just did one day

Question 55: Please rate the following: What level of benefit did MBSE bring to your project? Please elaborate.

Results (Average Rating = 3.89)



Open Ended Responses (Please elaborate.)

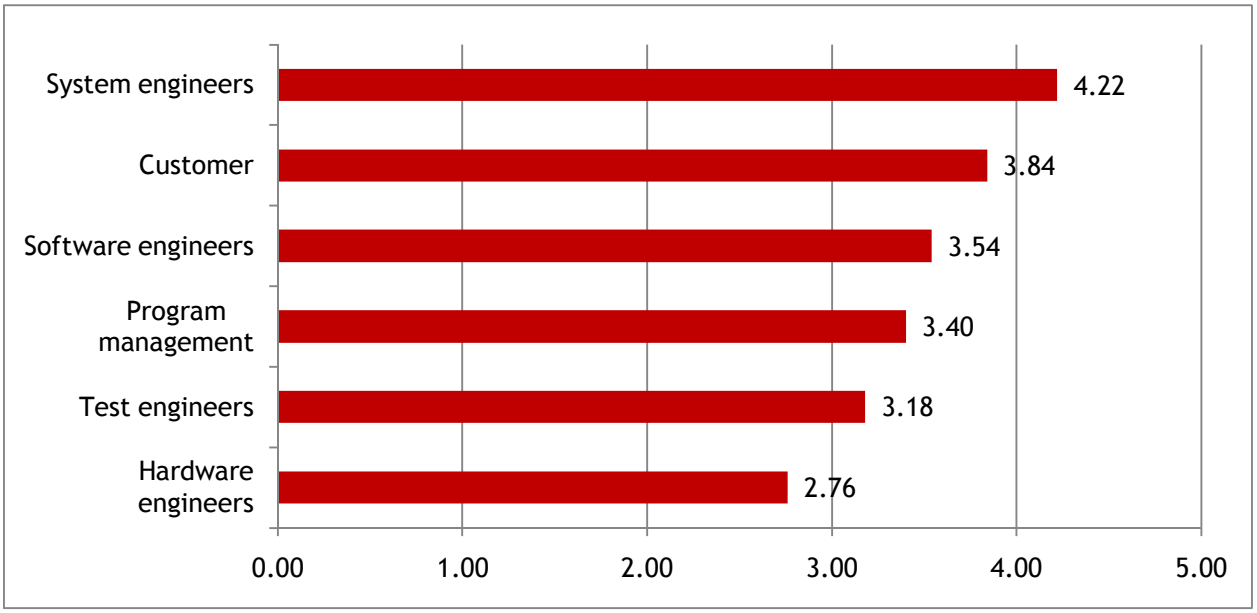
- What used to be a conflicted bag of mismatched illustrations, inconsistent prose and varying intent has become a structured model focused on needs and architecture.
- Politics was a constraint on effectiveness.
- Here, we need to define MBSE. The mathematical graph models and their associated hypotheses were critically important. The boxes and arrows much less so. What is a model? I'm not at all sure we are talking about the same thing.
- Still assessing.
- SysML is a notation only. To apply it we must follow a methodology.
- Considerably different - very useful. Things broke down though - with metrics, lack of Architecture modeling, I left
- Students were already familiar with MBSE from studying complex systems.
- The Systems Engineers did not want to use MBSE. They fought it because they did not see a value.
- MBSE is being adapted by the contractor for the entire program. We in the Govt Test community are using it for Test Enterprise Architecture. This looks to be the direction of the future with interoperability upgrades for DoD
- Shows the system engineering better than in a documentation approach.
- Finding great use with Use Cases combined with Activity diagrams. Seeing potential uses of IBD and BDD diagrams, but struggling with buildling a comprehensive model with all layers and allocations between layers.
- Remove ambiguity, speed up the maturity of the specification, better view of the whole, tackle inconsistencies, facilitate the communication between team members, fine grain traceability, automation of some tasks.

- MBSE is not yet used comprehensive in the system development.
- As pilot project with limited peer training, there have been challenges getting team members "up to speed" to the point of being proficient and productive. So the benefit to the project is limited but longer-term benefit to organization is (hopefully) more substantial (time will tell).
- Better system understanding enhances the communication and avoids misunderstandings in early concept and design phases.
- Useful R&D results and a beginning of relevant usage examples.
- Although initial experience is good it is too early to make a definitive statement.
- The most important point is that it provides a common way of expressing different systems.
- If my constituency had been more patient, it would have brought very high value; on the other hand, if the things I speak of above had been fixed, the constituency would not have run out of patience I believe
- We got more value from trying to organize our extensions of SysML in terms of maintaining parity across 3 views: - an ontological perspective where we can look at inferencing as a technique for analyzing a model (i.e., instances of the ontology in this case) - a metamodeling perspective where we can look at analyzing a model using OCL and QVT. - a profiling perspective where we can look at usability issues for tailoring existing SysML tools to make the ontology/metamodel user-friendly. In other words, if we want to use SysML as an infrastructure for domain-specific language extensions, then there is currently no specific methodology for developing DSL extensions of SysML and we had to make our own.
- By capturing behavior early, many complex details were identified earlier and at greater depth.
- The potential for high value exists, but instituting change on a large, complex, existing program is very difficult. Difficulty overcoming inertia and existing process limited opportunities for benefit.
- The use cases provided a fondation to discus specific behaviors with in the project.
- Much of our MBSE is still under development, particularly those elements which will effect and alter our design descisions. We have not yet established a complete enough model to contribute to project products.
- I had no idea what the end product should look like. Internal training completely changed that. Very useful
- The requirements we produced were of a higher quality than could have been produced without it.
- Only one person is using it - me.
- The model added clarity to the concept and requirements, but funding constrants curtailed the concept development & architecture effort. Analysis results & artifacts were rolled under a technology demonstration effort, which had a completely different emphasis, and so were ultimately underutilized.
- Building the model validated and codified the key element relationships, and began to tease out the temporal requirements.
- Again with an emphasis on domain knowledge. You can only model what you understand. Making meaningless models based on wrong understanding is a big waste of time. In fact spending time on methodologies before you even comprehend the problem is so common it is tragic.
- This is an unknown for us - MBSE was mandated by our customer, we're just giving our customer what he wants - the benefits to us are not tracked.

- has yet to materialize, except in one project. see the value, but too many issues to make work well until many members of team are trained and have bought in to process.
- Though the project was prematurely terminated, the early phase of system definition and use context was well received by peers. Our design was well understood with a level of engagement by peers below that which would be required from a document based method.
- It helps to level set a group of people and allow them to start being productive much earlier. Without it expect a lot of rework on model organization, best practices, model consistency and so on.

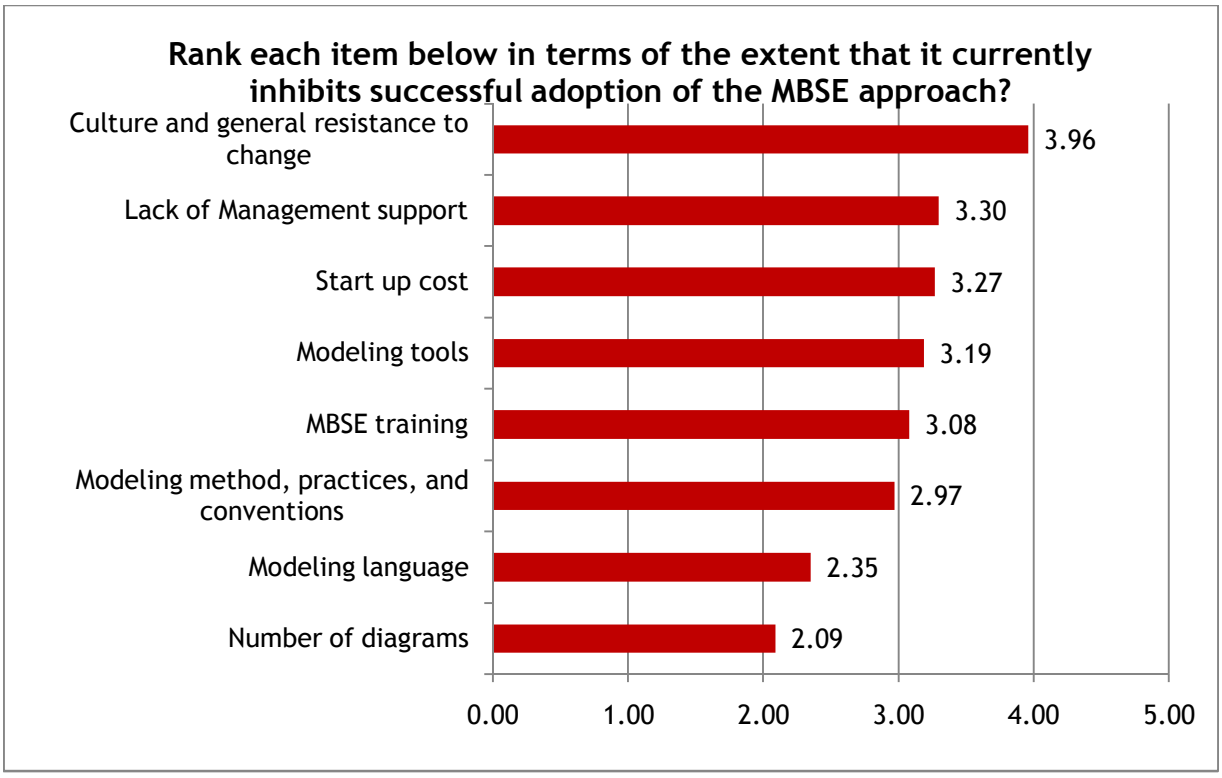
Question 56: How were modeling results perceived by the project stakeholders?

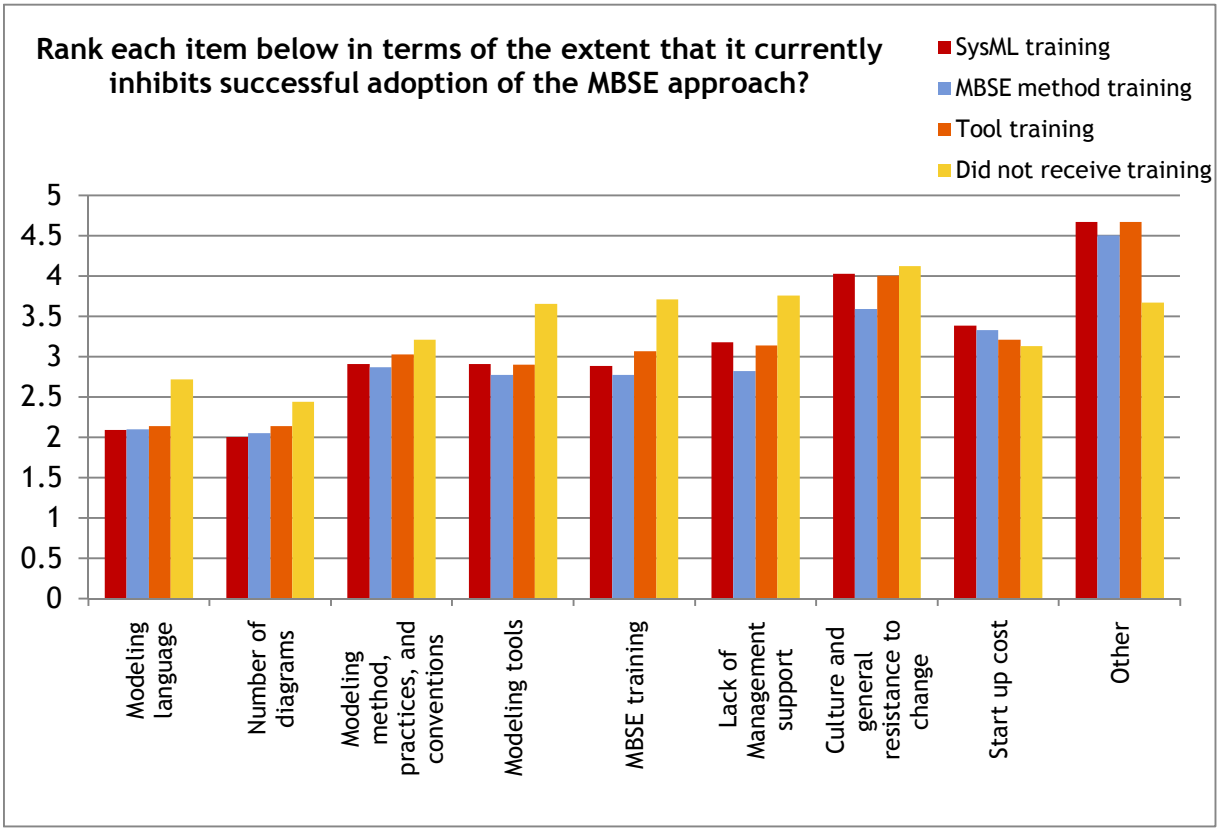
Results



Question 57: Rank each item below in terms of the extent that it currently inhibits successful adoption of the MBSE approach? Please elaborate.

Results


Rank each item below in terms of the extent that it currently inhibits successful adoption of the MBSE approach?
Culture and general resistance to change
3.96
Lack of Management support
3.30
Start up cost
3.27
Modeling tools
3.19
MBSE training
3.08
Modeling method, practices, and conventions
2.97
Modeling language
2.35
Number of diagrams
2.09
0.00
1.00
2.00
3.00
4.00
5.00


Rank each item below in terms of the extent that it currently inhibits successful adoption of the MBSE approach?
SysML training
MBSE method training
Tool training
Did not receive training
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
Modeling language
Number of diagrams
Modeling method, practices, and conventions
Modeling tools
MBSE training
Lack of Management support
Culture and general resistance to change
Start up cost
Other

Open Ended Responses (Other)

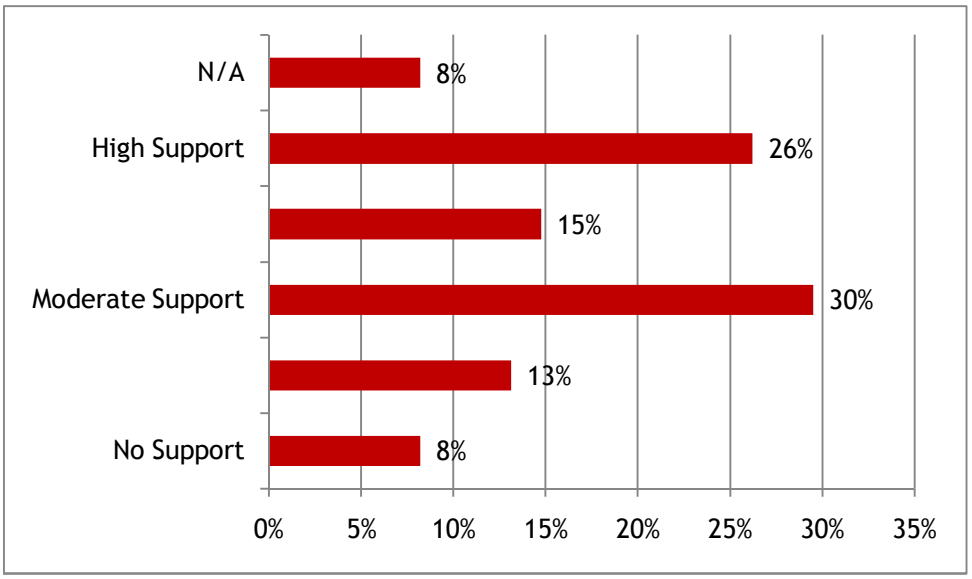
- The tool is the window that all of this stuff is viewed through. as such the quality of the tool is especially in terms of performance /productivity and maintaining the integrity fo the data is paramount. Systems engineers aren't used to the way software engineers and developers work.
- The doctors and biophysicists needed more training in the language. They were generally made aware of the formalism and foundations, but used the language at a rudimentary level.
- Model overall architecture with Objects
- There are fundamental underlying conflicts between an engineering perspective and a scientific perspective. At times, in practice these are almost mutually exclusive. Ultimately, whether this artificial boundary is broken is an open question.
- This has only been introduced from the grass-roots level. Current management wants to let developers do what they want to do (inspired by heavy emphasis on corporate LEAN). Trying to introduce anything that has a steep learning curve or that immediately strikes engineers as esoteric is a hard sell from a parallel position or distant position in the organization. When I have control of a project, I personally choose standards based approaches and modern best-practices that make sense. It is an easy sell to the team (they buy-in willingly--no coercion) if I (informed and motivated to do the right things) am the one in charge.
- SysML is a language, but the tool selected to elaborate SysML projects is more than important. Often, tools offer resistance to creativity and don't offer a guide to obtain SysML compliance in your project. In case you are introducing SysML: you need to learn the language (first effort), to select a tool (second effort) and to understand how the tool implement SysML (third effort, major difficulty). The feeling is to develop a SysML project without respecting formalism (tools permit to force SysML language outside the standards).
- The concept is new and is gaining momentum. The lack of online training for system engineers is the only thing holding back the widespread utilization. Things that help - Sandy Freidenthal's Book, the short course taught by Sandy for INCOSE-LA. INCOSE/OMG needs to make a series available online & via podcast.
- To introduce MBSE management support and continuous process improvement and user support is required.
- Inertia. Old school document-centric SE methodologies are well established in this organization and SysML/MBSE is still considered to be in the "early adopter" phase and not really on most SE's radar screen yet (my own work is helping to remedy that). Contracts with main customers are still document-centric (large formal reviews with long lists of CDRLs to be produced, etc.). The pilot project does have support of senior engineering though and I'm optimistic that more SE's will consider MBSE/SysML if the pilot proves to be successful and is well publicized.
- Our company systems are built around the document paradigm so to ensure acceptance we have made sure that system design description deliverables can be generated from the models. The inhibitors we hope to avoid with this approach are: - 'The legacy of the document paradigm' - we know how to manage and approve documents - Culture and general resistance to change The following applies to our methodology - We have minimized the number of diagrams used and use of fancy modeling constructs. KISS applies. - Keep ambition low - we only aim at improving the quality of our deliverables not to reap the all potential benefits of modeling.

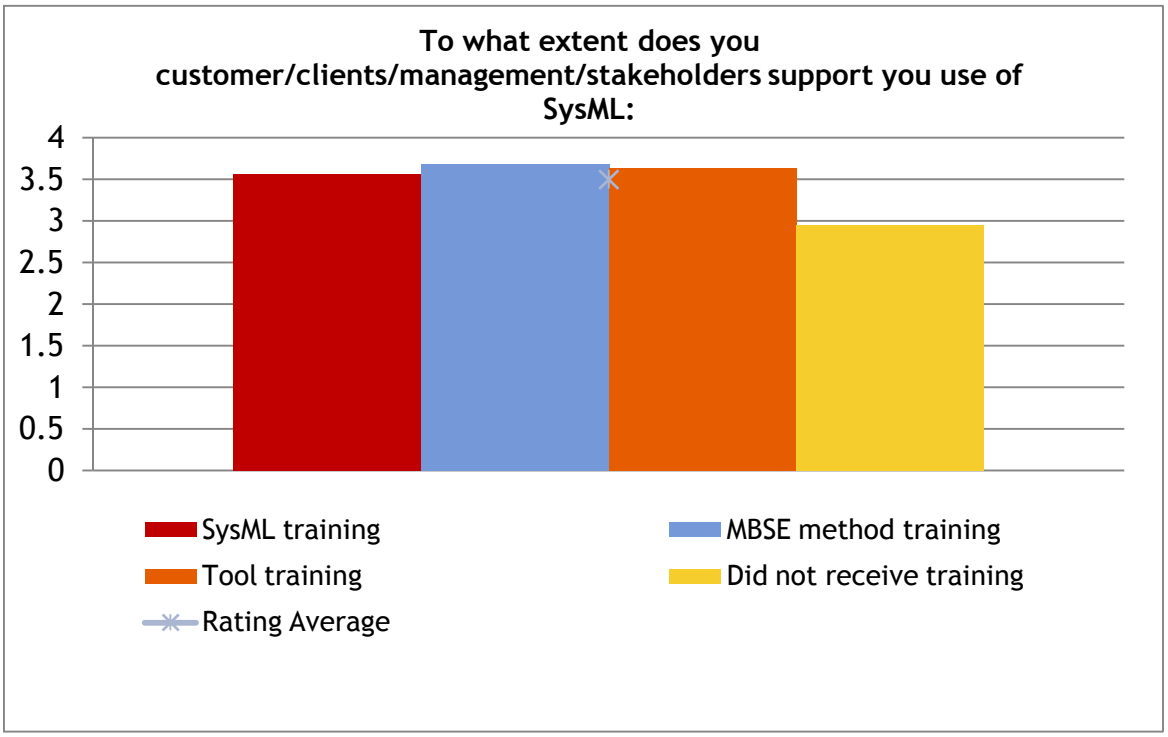
- If there were a modeling tool that you can learn to use like Microsoft PowerPoint, a model library containing domain-specific models of various problem domains, and a mechanism of drawing diagrams only in the way specified in a domain-specific model, everybody would start modeling.
- I distinguish between the modeling language, which is not necessarily an inhibitor, and the way it is presented, which is quite intimidating to just plain software developers (as opposed to modeling geeks or systems engineers)
- The "other" is that deploying MBSE in practice is a hard-sell because very few people understand what modeling is really about. To most, modeling is about making pretty diagrams. Even to experienced users, if we ask: "why is your project developing this (large) model?" -- the answer is often -- "to understand what we currently have" or some variant of that. What's lost is the potential for using a model as an explicit representation of what we know at some level of abstraction. We need to be able to use a model as an oracle and ask questions to that oracle as a proxy of our collective understanding of whatever it is we've modeled. Without this fundamental capability to query a model and compare the responses with expected responses or acceptance criteria, then it is very difficult to convince folks that continuous integration is an important part of agile & pragmatic modeling practices.
- The modeling team created over 200 Use cases for the single project.
- Learning the language is not that difficult, provided people want to learn it. Many Systems engineers seem to be resistant. The user interface is much more difficult than creating a power point diagram and without someone to wrtie code to manipulate the database there is little added value for a systems engineer. The methods for applying the tool are also immature, I have personally been involved in months of discussion about how to represent a MIL-STD-1553B Interface definition down to the bit level. This is very simply done in a table in word and not one questions it, ye tin a model it seems difficult. The tools are also not working at the same level of performance as Microsoft office, as a result many engineers are using power point and happy.
- Initial Project D progress was highly enabled by enthusiastic management. After merging with tech demo, emphasis shifted from abstract architecture to concrete implementation, and modeling results were not maintained.
- Distributed engineering/modeling on a single model image required discipline to avoid CM issues. Modelers were highly geographically distributed Lesson from Project D - no CM tool was used.
- It is easier/faster to create a flow chart in Visio and in many cases to write code than it is to implement an operation of a block as an activity diagram in the model due to the way the tools work (selecting stereotypes, providing types and directions on the pins etc)
- the investment in training and maintenance of the tool are the biggest cost drivers. the consistent usage of the model language was converting between straight functional engineering into OOSEM (more a blending)
- Our EE department is the biggest inhibitor on the hardware engineering side. They have their own tools, but fail to realize the value of performing the system design in a SysML/MBSE format to capture the requirements, test cases, etc.
- Old guard unable to move from structured analysis and design to object oriented. However, since SysML supports IDEF0 this shouldn't be a problem for the language itself. It's more of a project team issue.

- Models must be transportable between tools. Tool cost is high and upfront investment in training and system definition cost are not well received by program management. Need metrics that convey the value of MBSE.

Question 58: Please rate the following question: To what extent does your customer/clients/ management/stakeholders support your use of SysML? Please elaborate.

Results (Average Rating = 3.41)





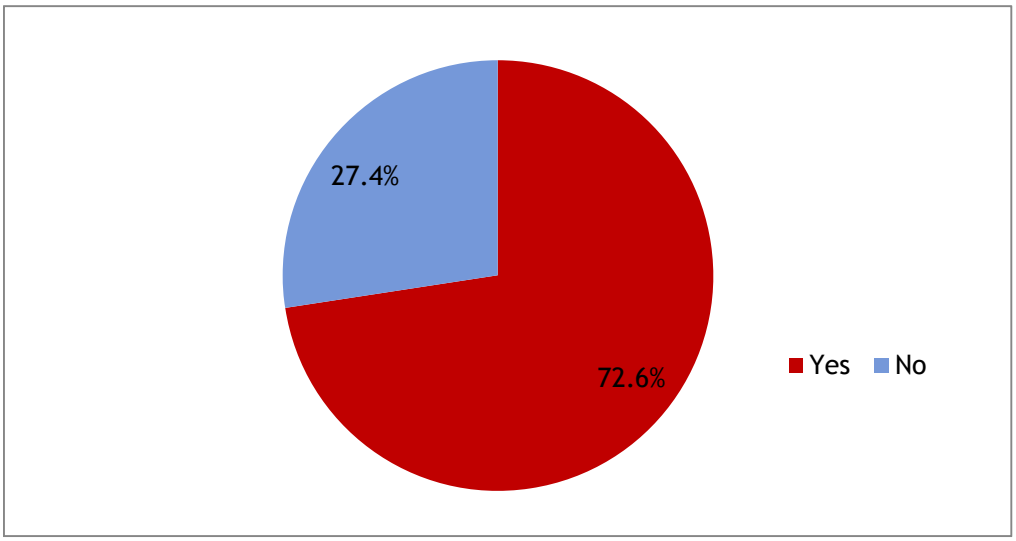
Open Ended Responses (Please elaborate.)

- Its something I have to manage the risk for. I have to nurse people into using it and prop up the tools when they falter. basically I minimize distractions and keep them focused on the power of the capability.

- It's hard to answer - the underlying formalism led to the break thrus.
- Organizational support was not problem. Changes due to budget was a problem
- Not because of its capability, but due to its potential.
- Our department tries to stay at the edge of the state of the art.
- They all want results, but one project has been able to continually justify giving us funding to continue, in part because we are doing all the right things (standards based approach) and have captured what we have done in a credible form.
- MBSE and SysML are gaining momentum in the Aerospace and DoD arena. The only drawback is training.
- Working with the Canadian Space Agency which has been an early adopter of SysML.
- Management: More costs without measurable return of investment.
- MBSE / SysML methods and tools are too immature to expect more than moderate support.
- Only low-middle management supported us in promoting SysML activities.
- If there were a mechanism of defining domain-specific models, everybody will support SysML.
- Support for SysML has gone underground in my organization ... I have a manager who's very supportive but knows that others don't want to hear about it anymore, at least for a long while ... so I am applying the concepts quietly
- MBSE is recognized as being strategically important and SysML is an important enabling specification for MBSE.
- Personal decision to apply SysML to effort as educational exercise. Customer & management didn't want to "absorb" time and effort associated with learning curve.
- Customer/clients/ management/stakeholders (some) usually are not interested in what kind of methodologies will be used for the project as long as the schedule and cost parameters are being met.
- Verbal support is in place. A full understanding of what such a cultural change really entails in investment of schedule, money, change to existing formal process, and delayed gratification has been lacking. This has led to significant frustration on the part of management and engineers.
- SysML and UML is only used when required. There is no encouragement for engineers to utilize this technology on their own.
- SysML is recognized as an emerging standard, but there is no guidance or preference to it.
- Initial support was enthusiastic, after merging with tech demo support became lukewarm.
- Scope chartered by corporate IRAD, by direction. Results mapped to a proprietary reference architecture, which was NOT developed in SysML.
- I think they are still waiting to see if this is going to be useful.
- The intent of the project was centered around the use of SysML so it had high visibility and support
- Did not perceive an appreciation of the value of MBSE outside the core system definition team.

Question 59: Would you agree to a follow-on interview regarding you use of SysML and MBSE (Note, We may or may not request a follow-on interview)?

Results



Question 60: This space is provided to expand your answers to question 5 regarding Diagram types.

Open Ended Responses

- All of my issues relate to correspondence. I don't think SysML has to many diagram types. I do think that achieving correspondence (while much easier than without SysML) could still be tightened dramatically. seq and activity can achieve different subtleties - however they have much in common or could easily see how a tool could toggle between representations. incorporating parameters (as in form parametrics) into activities is not straightforward either. or seqs for that matter.
- There seems to be a lot of focus on diagram types. This, like other elements of the SysML language should be extensible.
- Talk to Steve Janiszewski and Six Sigma, Inc.
- I think I added enough detail before to get the conversation started. Further elaboration probably best done in person.
- IBDs: We use SysML model to support generation of ICDs that look/read like the ones we typically produce using standard methods. We define complex interfaces between communicating parties that decompose into numerous services and further into specific info flows exchanged within the context of those services. We used a standard port to represent the interface with lower-level standard ports to model constituent services with further lower-level flow ports to model specific service message exchanges. We could not delegate those lowest-level flow ports to internal parts without validation warnings. Instead, the tool (MagicDraw) insisted on considering the service ports and the internal parts being interconnected as peer parts and so imposed requirement to conjugate one end. Even when defining the connector as a "Delegate" type of connector (a MagicDraw extension?) it still threw a validation warning about the flow direction mismatch. Perhaps SysML spec needs to be clear on how modeler should be able to enforce a delegation construct even when perhaps its not obvious. Perhaps this requires defining some notational convention to differentiate delegation connectors from assembly connectors? SDs: As part of standard, gradual definition one might want to initially define an interaction between the system and its external entities, then decompose the system into subsystems, and then further expand the interaction to show how the subsystems are involved. It would be nice to be able to keep working with the original sequence diagram and be able to toggle the scenario between the original view showing only the system and the deeper view showing the system boundary and its internal subsystems (something like expanding/collapsing in a tree view only applied left/right to lifelines rather than up/down). If this is part of the language then MagicDraw doesn't support it. - Perhaps the language needs to be clear on how one can interconnect reference interactions in a single sd? For example, should I be able to flow the outputs from one ref interaction as input to another reference interaction in the same sd? If so, MagicDraw doesn't support it. - Should be able to give a ref interaction a name that is specific to the context in which it is being referenced and have that name visible on the sd. Similarly should be able to give a lifeline a name that is meaningful within the sd that the lifeline appear in and have that name visible on the diagram. Validation: Perhaps this is a tool-specific comment. MagicDraw at least allows modeller to place any sort of item flow on a connector connecting 2 flow ports without throwing a validation warning if the flow does not match the flow spec. I would have thought this is a fundamental validation rule that every tool should apply by default but apparently it isn't?. I'd like a way to explicitly associate a message and its conveyed information in an sd diagram with

a specific connector in an ibd so that tool could raise validation warning if the two constructs don't match in terms of flow type and direction. Am I missing something in the language that supports this kind of cross-cutting linkage?

- Activity diagram A view of deeply nested activities is terribly missing. The same structured, "look inside" view, as in IBDs, would help enormously in avoiding having too many diagrams. --TimW 20:25, 3. Nov. 2009 (UTC) Nested activities are not allowed. Is it right that you mean CallBehaviorActions? The abstract syntax and semantic is different to ibds where we have a real nesting. Rkarban Yes, CallBehaviorActions. The syntax and semantic might be different but for visualsation reasons it would be much better to show different levels of behavior in one diagram, like in IBDs where you can reveal the structure of nested parts. Sometimes you have only three or four actions, where the call behavior of one action is important but you need to look at a separate diagram either in printed documentation of by navigating in the model. To keep track of the nested behavior is difficult. Great difficulties in keeping parameters, parameter nodes, and pins in sync. In particular for the <<rate>> attribute. --TimW 20:25, 3. Nov. 2009 (UTC) What is the specific problem with the rates? Rkarban SysML only provides only <<rate>> stereotype which extends only Activity Edge and Parameter. There is no <<rate>> for parameter nodes and Pins. From a users point of view it makes sense to have rate everywhere because you want to show the rate of a parameter on the pin of a call-behavior action without "digging" into the activity. Consistent stereotyping of Pin, Parameter node, and Parameter is not defined. --TimW 20:25, 3. Nov. 2009 (UTC) Do you mean that a parameter node and a corresponding pin must have the same stereotype? That makes sense from the user point of view, but the concepts are loosely coupled and the UML/SysML metamodel allows that they have different properties. Rkarban see above. In daily practice you have to change the rate at three different places which is cumbersome and error prone. Apart from that rate is not defined for pin and parameter node - see above. More than the <<rate>> attribute is needed for control systems, e.g. duration, complexity, latency, jitter, clocked, etc. --TimW 20:25, 3. Nov. 2009 (UTC) Did you check whether MARTE offers these properties? I agree that they are necessary, but I don't think that they should be part of SysML, but part of a profile like MARTE. Rkarban agreed. I think the "UML Profile for Modeling QoS and Fault Tolerance Characteristics and Mechanisms" provides such mechanisms. Instead of letting SysML define its own (yet another) rate it should see what it can re-use from existing UML profiles. I think instead of having simply a rate property one could define QoS for Parameters, Parameter Nodes, and Pins and use the definitions of the UML profile. Inheritance of parameters is not clearly specified. e.g. if new parameters should go at the beginning or the end of existing ones. Furthermore, the situation is complicated by parameters, pins, and parameter nodes which creates a hell lot of work for tool-vendors. Rkarban Activities and Parametrics are two sides of the same coin. The former describes behavior in causal manner, the latter in a-causal manner. When describing the behavior of a system the appear closely related and you would like to constrain attributes/parameters of actions/activities together with static system properties. Rkarban Block definition diagram Better description needed on how constraints on blocks relate to constraints in parametrics --TimW 20:25, 3. Nov. 2009 (UTC) There is no relationship. Rkarban even worse... Since their is no relationship it is sometimes quite ambiguous if you should describe a constraint on a block with parametrics or "simple" constraints. Often they are related or even the same. Better description needed for the difference between a normal part property and a part property with an association is. --TimW 20:25, 3. Nov. 2009 (UTC) Couldn't follow this. Rkarban A block can have a part property

with and without association relationship. Adding an association adds for example the possibility of navigability, naming the association, and have roles on either side of the association. Value Types are weird as they need Unit and Dimension but Unit also has a Dimension attribute. And, as value types have a dimension and unit, they can have themselves attributes which are value types - how does this fit together? --TimW 20:25, 3. Nov. 2009 (UTC) Good point. Fortunately that part of SysML was enhanced for SysML 1.2 and will be for future versions. I'll send you a paper about that. Redefinition of Property-specific types are a hassle as the COMPLETE hierarchy has to be redefined (inherited) to change the definition of ONE single deeply nested part. Redefinition of default values of value attributes in a specialization hierarchy is cumbersome (redefined property). The same attribute has to be created in every specialization and then redefined. When defining a new default value in a specialization the value is also changed in the base class. --TimW 20:25, 3. Nov. 2009 (UTC) Agreed. It is difficult for people to see what the difference between allocating an action/activity to a block and a block operation is. --TimW 20:25, 3. Nov. 2009 (UTC) Do you mean for example "ActivityA allocated to BlockA" versus "ActivityB allocated to OperationB"? I think it more a methodology issue than a SysML issue. Or do you have a proposal how to solve it? Rkarban I mean: "ActivityA allocated to BlockA" versus "OperationA of a BlockA". What is the difference? Could, in a certain methodology, an ActivityA be allocated to a BlockB and then again allocated to an OperationB of BlockB? Internal block diagram No proper syntax and semantics for delegation/junction ports. It must be possible to hide the internal parts of a part in an ibd without loosing the visual information that they have connections to the outside of the enclosing part. --TimW 20:35, 3. Nov. 2009 (UTC) Agreed. I've added some text. No proper syntax for discipline specific interaction points, i.e. ports for mechanics, optics, electronics, information --TimW 20:35, 3. Nov. 2009 (UTC) I think that's an issue for profiles. Difficult use of context-specific values due to dependency on UML InstanceSpecification. Some tool vendors help a lot, e.g. NoMagic with their implementation. However, there a different interpretations of the spec by different vendors. --TimW 20:35, 3. Nov. 2009 (UTC) Agreed. Unclear or no description of nested ports in order to group ports, e.g. group together all electronic, all mechanical, all optical, and all information interaction points in ONE port for better re-use and maintenance. --TimW 20:35, 3. Nov. 2009 (UTC) Agreed. Flow Properties CANNOT be connected individually but only the whole port --TimW 20:35, 3. Nov. 2009 (UTC) Agreed Package diagram The semantics of Views and Viewpoints is very vague and weak. --TimW 20:35, 3. Nov. 2009 (UTC) What is missing? Rkarban Maybe some more examples in the spec. The Viewpoint describe the used "language" simply as free text. In all the examples there is exactly ONE view related to ONE viewpoint. It's unclear how they relate. Is it a definition/usage relationship? Are the views are a kind of usage of the Viewpoint? I would like to see an example with more than one view conforming to a viewpoint. And most important, it is difficult see what the difference of a normal package and a view is. Practically there seems no difference. It seems only a theoretical difference. Parametric diagram It is not clear why a diagram different from an IBD is needed. They show practically the same things. Rkarban People have difficulties to see the difference between behavior described with activities and dynamic aspects of the system using parametrics, which act on static system properties. --TimW 20:35, 3. Nov. 2009 (UTC) Do have a proposal for that? I would define activities as behavior that is directly performed and managed by the system. Parametrics is not a behavior, but a network of constraints that must be valid all the time. That network is not managed by the system itself. Rkarban I attach an elaborate example (explanations are in the speaker's notes) from the telescope showing that they are very closely related, the two sides of the same coin. JPL ran into the same problem. The ppt and pdf are also in SVN.

Datei:ActivitiesAndParametrics.pdf Properties of Actions and Activities difficult to integrate, e.g. duration of an action could be bound to parameter --TimW 20:35, 3. Nov. 2009 (UTC) Do you propose that the duration of an action should be mapped to a parameter of the activity? Rkarban see attached example above. I propose to be able to use parameters/attributes of activites also in parametrics, and to be able to stick constraints to actions to bind paramters and attributes of actions to constraint parameters. Requirements have no attributes and can therefore not be bound to parameter. This is very important for a seamless model. --TimW 20:35, 3. Nov. 2009 (UTC) Agreed. Rkarban Requirement diagram Requirements shall have attributes in order to bind them to parameters in parametrics models. --TimW 20:35, 3. Nov. 2009 (UTC) Isn't that the same as above. Rkarban indeed. Rkarban Sequence diagram If a Call-Behavior of an Action is a Sequence diagram it is not at all clear how object flows and control flowsare handled in the Sequence diagram. --TimW 20:35, 3. Nov. 2009 (UTC) More general it is not clear how to handle object flows at all. The interaction models describes the exchange of messages. Sometimes it would be useful to use allocated parts in sequence diagrams, e.g. when SW is allocated to HW one would liketo use also the SW components in the sequence diagram. --TimW 20:35, 3. Nov. 2009 (UTC) In which sequence diagram should the SW component appear? Rkarban in the sequence diagram of the block to which the parts/blocks have been allocated. Rkarban State machine diagram none Rkarban Use case diagram none Rkarban Other features (e.g. allocations) <<allocation>> should not be a dependency as the direction of the dependency relationship is not always clear, e.g. when deploying software blocks to hardware blocks the software does not really depend on the hardware nor the other way round. The usage of <<allocation>> in definition->definition and usage->usage is not clear but this might be a methodology issue. The spec must be clearer about allocation schemes of usage->usage, definition->definition Currently the swimlane can also represent a block but when an action w/ a call-behavior is allocated what should happen? How should the action and call-behavior be allocated correctly? as an action needs the context of an activity. Allocation ObjectFlow to ItemFlow: The ObjectFlow (Edge) describes that in the context of an Activity the output of one Action is bound to the input of another action. In the context of a block a item flow describes the flow of an object from one part or port to the connected part or port. The allocation of the ObjectFlow to an ItemFlow defines which ObjectFlow corresponds to which ItemFlow in a given context. Supplier and producer and context need always be defined. Allocation strategies depend very much on the method used. The difficult part is to use them properly, e.g. the chain: Use Case -> Behavior -> Structure Allocation is a stereotype of UML abstraction and the semantics (i.e. the exact mapping) of allocate are not defined in SysML. Allocation Pin to Port not addressed in SysML standard 1.1 SysML only provides only <<rate>> stereotype which extends Activity Edge and Parameter. Some tools extends these stereotypes to ObjectNode Synchronization of Parameter and Pin is tool-dependent.

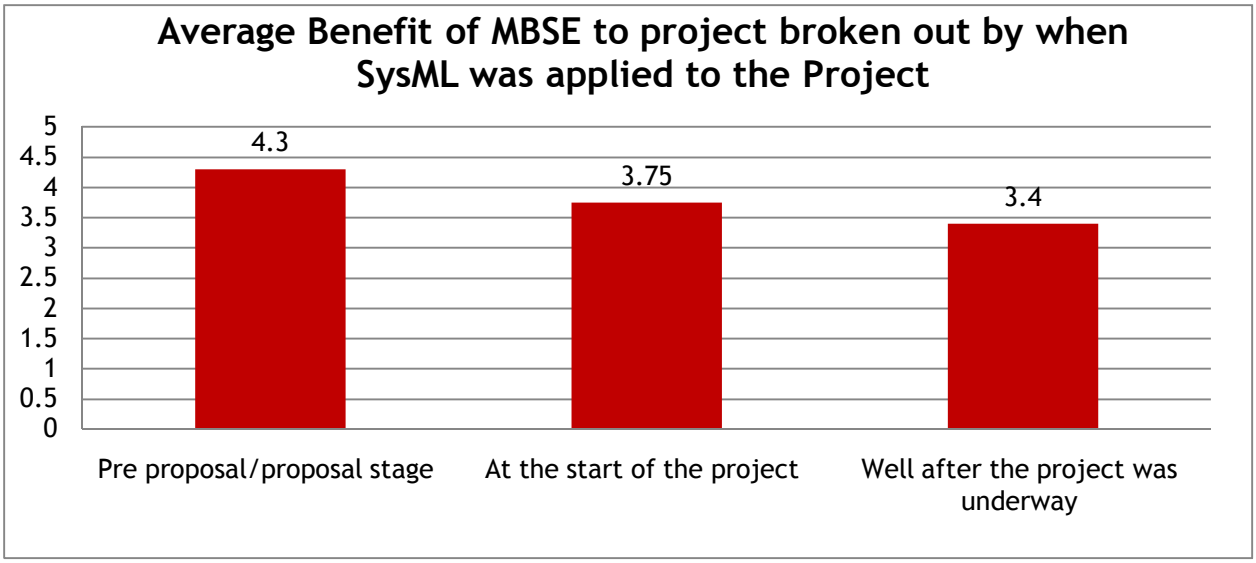
- For example, communication protocols can be expressed with SysML blocks. Any communications protocol has the following types of ports but they don't need to have other types of ports. - direct ports for other protocols to access them, - direct ports for them to use other protocols, - indirect ports to interact with peer protocol entities. If SysML had a simple mechanism to define <<protocol>> that can have <<access-port>>, <<use-port>> and <<peer-port>> but no other ports and if there were a tool to draw protocols with only those port types, it would be easier to express protocols with SysML and to draw diagrams of protocols. What is required may be a mechanism to define viewpoints in a formal way.

Question 61: This space is provided to expand your answers to question 15 regarding recommendations for SysML diagram types. If you want to maintain a hard copy of this, please print this using your browser print function before proceeding.

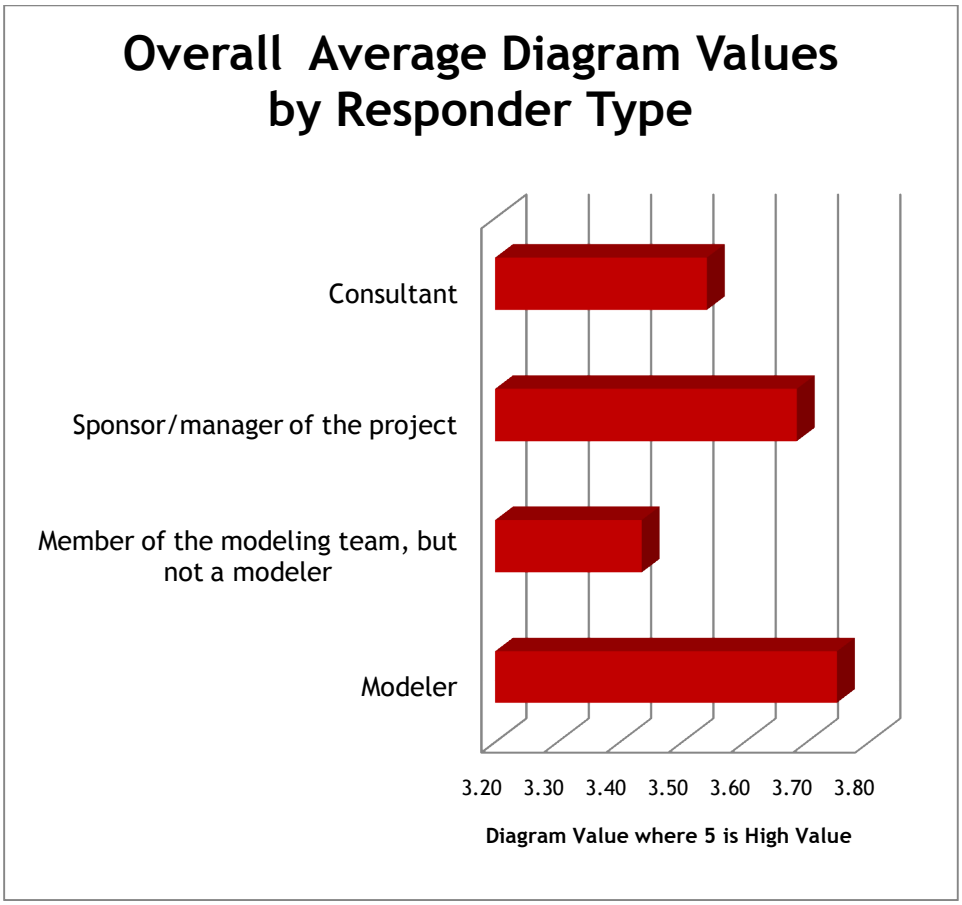
Open Ended Responses

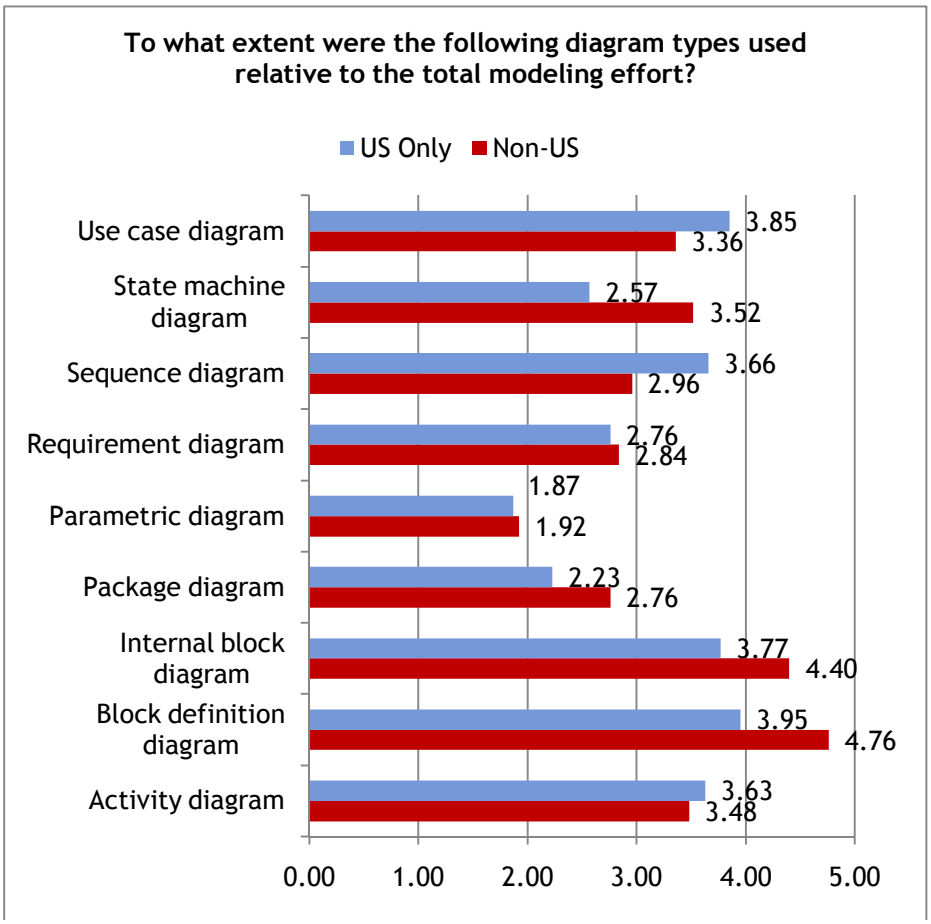
- Timing diagrams would be useful. a better representation of time throughout the model would be way better.
- We used SYSMOD, but there is no orientation from SysML. Should SysML contain some orientation about the methodology (MBSE) to be used?
- I am done
- Micouin's Theory about requirements has been published in the Systems Engineering journal. Reference and abstract can be found here: http://www3.interscience.wiley.com/journal/117951435/abstract?CRETRY=1&SRETRY=0
- See Q60 for some suggestions

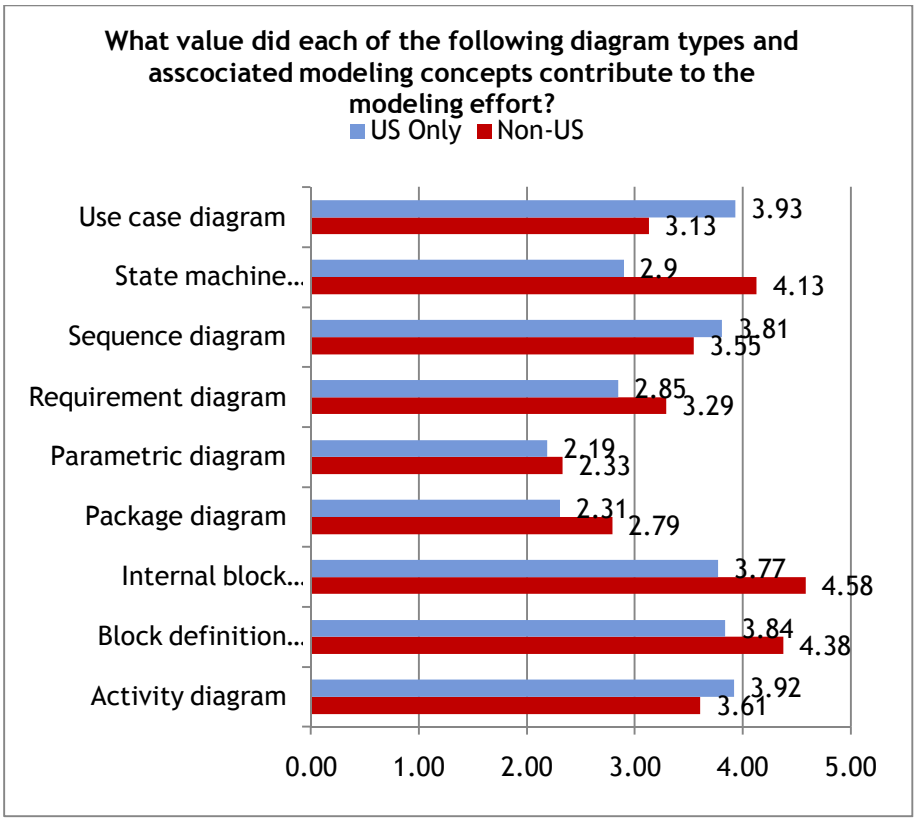
Cross Correlated Results

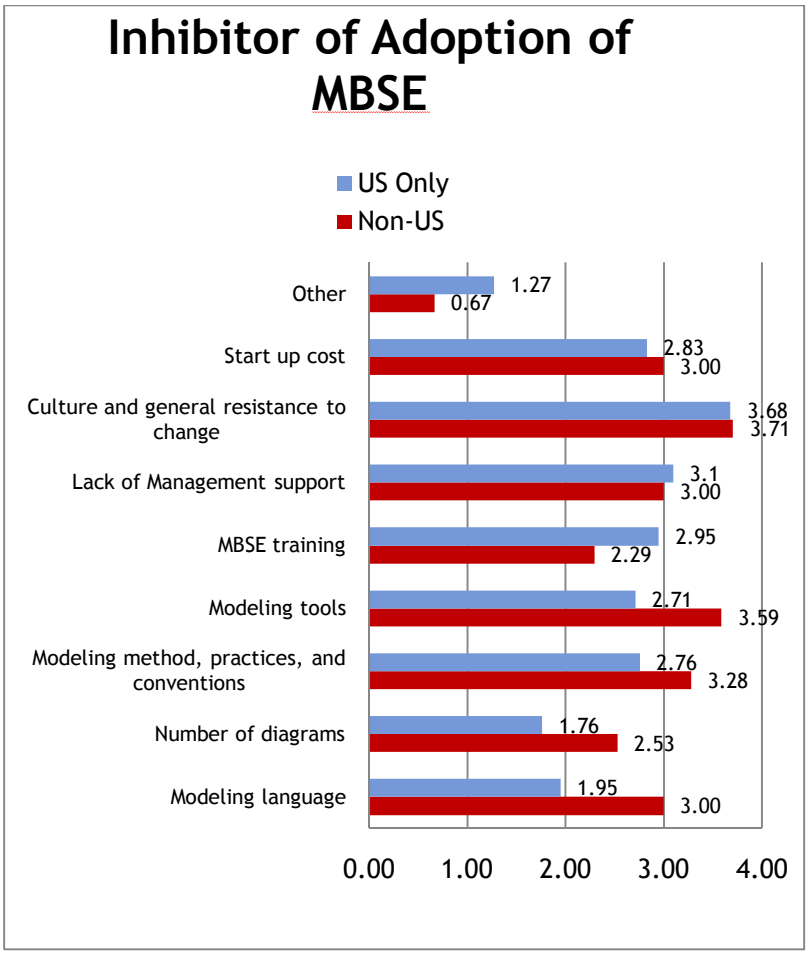


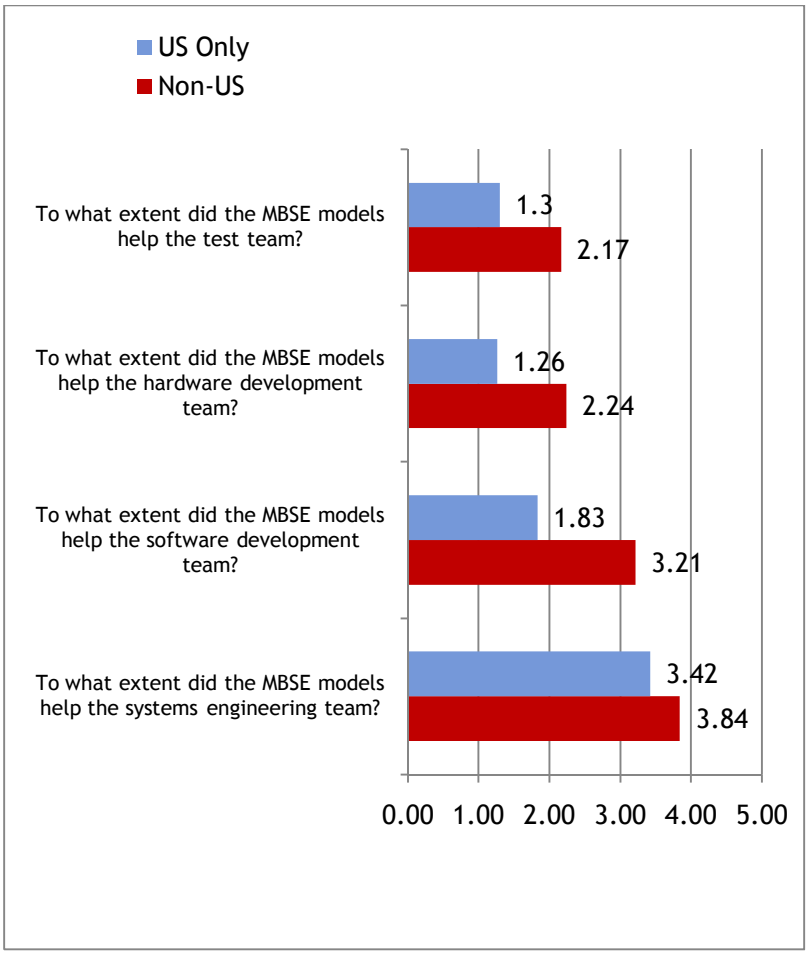
	Artisan Studio	IBM Rhapsody	InterCAX ParaMagic (MagicDraw plugin)	NoMagic MagicDraw	Sparx Systems Enterprise Architect	Rating Average
Overall Value Average of all Diagrams	3.88	3.57	4.21	3.95	3.87	3.82

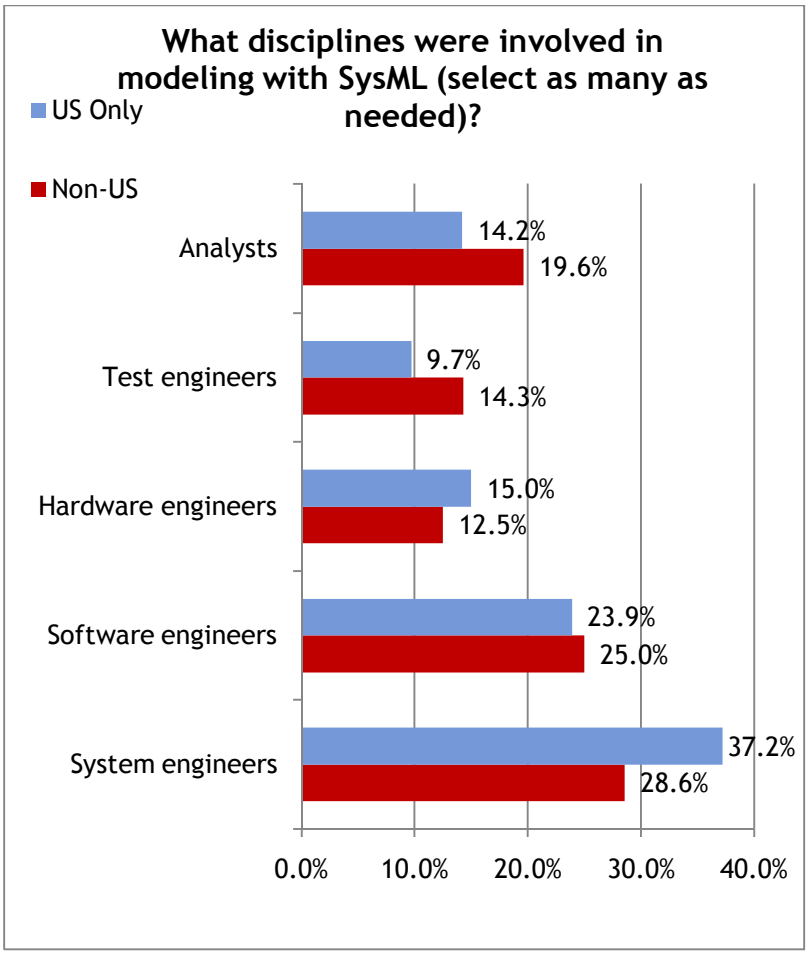

Overall Average Diagram Values by Responder Type
Consultant
Sponsor/manager of the project
Member of the modeling team, but not a modeler
Modeler
3.20
3.30
3.40
3.50
3.60
3.70
3.80
Diagram Value where 5 is High Value

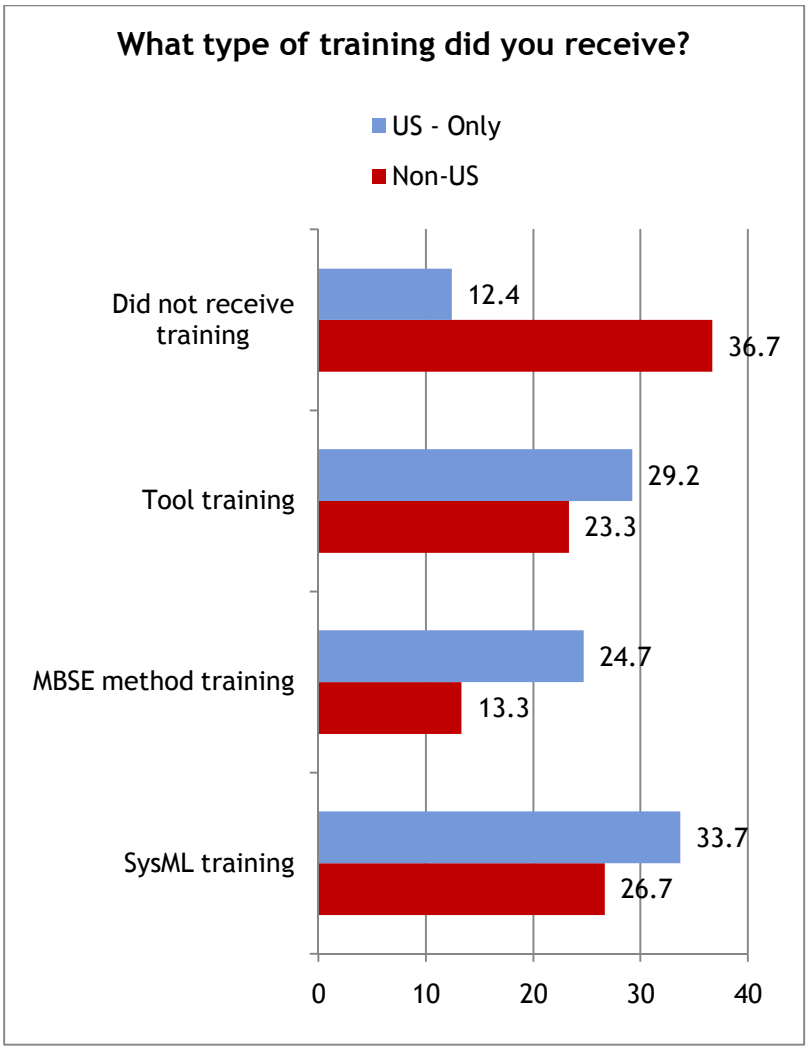

To what extent were the following diagram types used relative to the total modeling effort?
US Only
Non-US
Use case diagram
3.85
3.36
State machine diagram
2.57
3.52
Sequence diagram
3.66
2.96
Requirement diagram
2.76
2.84
Parametric diagram
1.87
1.92
Package diagram
2.23
2.76
Internal block diagram
3.77
4.40
Block definition diagram
3.95
4.76
Activity diagram
3.63
3.48
0.00
1.00
2.00
3.00
4.00
5.00


What value did each of the following diagram types and asscociated modeling concepts contribute to the modeling effort?
US Only
Non-US
Use case diagram
State machine...
Sequence diagram
Requirement diagram
Parametric diagram
Package diagram
Internal block...
Block definition...
Activity diagram
3.93
3.13
2.9
4.13
3.81
3.55
2.85
3.29
2.19
2.33
2.31
2.79
3.77
4.58
3.84
4.38
3.92
3.61
0.00
1.00
2.00
3.00
4.00
5.00


Inhibitor of Adoption of MBSE
US Only
Non-US
Other
1.27
0.67
Start up cost
2.83
3.00
Culture and general resistance to change
3.68
3.71
Lack of Management support
3.1
3.00
MBSE training
2.95
2.29
Modeling tools
2.71
3.59
Modeling method, practices, and conventions
2.76
3.28
Number of diagrams
1.76
2.53
Modeling language
1.95
3.00
0.00
1.00
2.00
3.00
4.00


US Only
Non-US
To what extent did the MBSE models help the test team?
1.3
2.17
To what extent did the MBSE models help the hardware development team?
1.26
2.24
To what extent did the MBSE models help the software development team?
1.83
3.21
To what extent did the MBSE models help the systems engineering team?
3.42
3.84
0.00
1.00
2.00
3.00
4.00
5.00


What disciplines were involved in modeling with SysML (select as many as needed)?
US Only
Non-US
Analysts
14.2%
19.6%
Test engineers
9.7%
14.3%
Hardware engineers
15.0%
12.5%
Software engineers
23.9%
25.0%
System engineers
37.2%
28.6%
0.0%
10.0%
20.0%
30.0%
40.0%


What type of training did you receive?
US - Only
Non-US
Did not receive training
12.4
36.7
Tool training
29.2
23.3
MBSE method training
24.7
13.3
SysML training
33.7
26.7
0
10
20
30
40