Piloting Model Based Engineering Techniques for Spacecraft Concepts in Early Formulation

Bjorn Cole
Mission Systems Concepts Section
Jet Propulsion Laboratory,
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109-8099

Chris Delp
Flight Software and Data Systems Section
Jet Propulsion Laboratory,
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109-8099

Kenny Donahue
Mission Systems Concepts Section
Jet Propulsion Laboratory,
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109-8099

The JPL Integrated Model-Centric Engineering (IMCE) project is tasked with bringing Model Based Engineering methods, tools and practices into use by systems engineers. Taking a Europa-focused flagship concept proposal as an example of a system that is early in its engineering lifecycle, IMCE piloted architectural elements for an integrated modeling environment. These elements were used together both as a proof-of-concept and to guide development of the methods, tools and practices for flight projects as JPL transitions to model-centric engineering. This paper presents both products of this effort and a variety of insights to guide the future of model-centric engineering techniques. Also, recommendations are made regarding the strengths and weaknesses of existing information models and the Systems Modeling Language (SysML).

Introduction

The purpose of this paper is to describe the Integrated Model-Centric Engineering pilot performed using data from the Europa proposal for the 2008 NASA/ESA Outer Planets Flagship Mission study. The pilot was based on this mission concept because it is still in early development, and it is a prime example of the trend toward increasing ambitions in space science and its supporting flight systems. The pilot addressed multiple classes of modeling and integration problems, and provided proof-of-concept solutions where possible. Where model-based tools are still immature, issues were identified. The intention is threefold: to pilot ways to move conceptual design information into a model-based platform; to understand how this information can provide a solid foundation for scaling; and to reduce the gap between SysML concepts and the conceptual patterns of space systems engineers at JPL. The pilot produced a series of insights into general integration issues, the strengths and limitations of common modeling, simulation, and solving tools, and suggestions on how to ease the learning curve for systems engineers used to working in a conventional regime.

The Object Management Group's Systems Modeling Language (OMG SysML) (Object Management Group, 2008) was used as the foundation for an information model of the Europa concept. A variety of architectural views, models, and alternate representations of the design were generated in this language. Some basic simulations of various spacecraft

behaviors were used to introduce dynamic analyses. These were built in conjunction with existing executable trade models. As the team of authors became satisfied with the quality and fidelity of the representations of the spacecraft, they presented these results to systems engineers working on the project. The results of these interactions, which occurred during and after this effort and during its follow-on, are the source of the insights presented in this paper. A number of these should be taken to heart by both developers of the SysML language and tools, simulation tool makers and those that are pushing for adoption of model-based engineering.

Background

Systems engineering is a well-established discipline with a large body of work, both practical and theoretical, concerning the problems of describing and controlling designs. Collectively, the current set of approaches has been labeled a “document-centric” practice. This is due to the fact that design control is executed by creating legalistic natural language statements, which are the current form of design requirements. The basis of these requirements are examined and reviewed for their rigor and appropriateness through a series of formalized meetings, which use documents that specify both the content and the structure of the review. Further, a great deal of day-to-day design work is conducted and communicated by passing around emails, diagrams rendered by software or hand, and technical memos. There are a number of issues that have been raised with the current practice, including questions of the accuracy of description, burden of document maintenance, and the tendency of static information to languish in a repository. These are the issues that are often highlighted by advocates of a new approach.

Static documents have a number of drawbacks when considered in a world of modern information technology, but they are created to contain the kinds of information needed to move from a concept to a design to a spacecraft, and then how to operate it. The ad hoc documents created from day to day represent the types of information subsystem and domain engineers need to communicate with each other and the systems team. Formal documents, as encoded in the NASA Systems Engineering Procedural Requirements (NASA, 2007), trace out a series of way stations for the system design to reach on its way to construction and operation. In sum, these documents embody information with an established need.

Model Based Systems Engineering (MBSE) has a long history as well (Wymore, 1993) but has never fully taken hold within the systems engineering discipline. While earlier work has focused on developing mathematical theory for MBSE, current computer aided Model-based approaches invoke the CAD paradigm that has found success in other engineering disciplines as a way to visually construct complex information models. Thus, MBSE contrasts with document-based approaches by stating that they aim to develop the design through model elaboration rather than document elaboration. There are a wide variety of MBSE processes, including SYSMOD (Weilkins, 2008), OOSEM (Friedenthal, 2008), a systems adaptation of the Rational Process, and JPL’s Systems Analysis (Estefan, 2008). These processes each follow the general outline of defining the use of a system, its conceptual structure, design down to blueprints, build, integrate, test, and operate. Some discuss disposal as well. But most of these end up emulating lifecycle and processes that already exist, using models as the major artifacts of those processes.

The proposed value of Model-Based Systems Engineering (MBSE) is the ability to describe the structure and behavior of an integrated system according to the concerns of stakeholders rather than as a collection of subsystems. This is no different than systems engineering in general. However, the problem with most complex systems is that of integrating information

across many large teams. As designs grow and change, ripple effects develop and maintaining documentation becomes a Sisyphean task. This is compounded by the fact that increasing use of models in both systems engineering and other engineering disciplines produce an information overload that exceeds the ability of office productivity tools and artifacts to track. Alternatively, graphical standards based descriptions rendered from a common repository of information can provide the richness of information provided by computer-aided design tools within other disciples. They can also serve to improve design integration and system-level performance. Technologies from the semantic Web (Herman, 2009) can be used to run advanced queries against formalized models. This is balanced against the concern that the information systems and formalisms needed to implement MBSE will divert the attention of the systems engineering team. MBSE is about making the practice of systems engineering as effective as possible.

Modeling Concerns in Formulation and Phase A

In the early phases of a design, problems of information do not present themselves in terms of complexity of the design or team. The team is small, and so communication is straightforward. The technical information to be communicated is relatively abstract – families of options and general design principles rather than the connections between specific pieces of hardware. Problems of information do appear if the tempo of design increases or if there is an interest in automated search methods. At its ideal, early phase design should be investigating a wide range of design alternatives and be very open to change. The questions of data exchange are less about keeping a large team in the loop as they are about gathering and absorbing data about a wide variety of options, and transforming them into a common basis so that they can be compared fairly and rigorously. The driving questions are “what is different between these concepts?” and “what changed when we looked at this issue?” The study and elimination of design alternatives is an important aspect of the rationale used to support the choice of a given system architecture. An example of this is in an appendix to the Europa concept report (NASA, 2008).

NASA follows a standard process for approving a mission as its technical and managerial basis matures. The early formulation phase typically involves moving toward a proposal for either a mission or a spacecraft to fulfill a directed mission. The approval of the proposal marks Key Decision Point (KDP) A. At this point, a new pot of funding is made available to progress to further formulation and the point of ultimate decision as to whether a project will move forward: KDP-B. Just after both of these major points, the resources and the need for an increase in project personnel arrive. It is at these points that a clear and comprehensive description of the design is needed to quickly integrate the new staff.

Early formulation is primarily concerned with developing a feasible concept with realistic resource (schedule, cost, risk, etc.) budgets that can be used to compete with alternate concepts. Once KDP-A is reached and the project moves into Phase A, the attention turns to understanding how the mission will actually be achieved. At this point, the value of modeling and MBSE greatly increases, since it makes the implicit explicit and highlights design issues. The focus of the work in this paper is primarily Phase A and beyond.

The value of a model-based approach then evolves into those given earlier: scalability and accessibility of information in a central repository, a touchstone of truth, and the ability to retrieve information that otherwise requires scanning through multiple documents. These capabilities are desired to help with the increasing challenges of managing a maturing design. Decisions made early in design manifest their strengths and weaknesses. Changes in design balloon in cost and difficulty.

A model-based approach aims to address these challenges by bridging design reviews with superior artifacts, automatically propagating design changes, speeding simulation in support of decision-making, and maintaining design integrity through to operations and retirement. The effort is aided by starting early. When the design is young, the information needed to describe it is manageable, both in terms of modeling effort and the cognitive load it applies to its system engineers.

Integrated Modeling Architecture

The desired outcome of this task was to demonstrate a proof-of-concept implementation of architecture for integrated modeling. The architecture provides a "single source of truth" contained in an information model. This is capable of integrating executable simulations as well as providing the ability to generate up-to-date, error-checked specifications. In this way, project systems engineers and domain systems engineers would be able to construct an integrated design and build a specification set for spacecraft projects that would possess the quality attributes of scalability, manageability, durability, simplicity, communicability and usability.

The core of the architecture is a standards-based information model that will be durable across the lifecycle of the project. For the pilot, this architecture was centered on SysML to make diagrammatic depictions of the design. No Magic's MagicDraw implemented the information model in its software, and InterCAX's ParaMagic was used to attempt to integrate the analysis and simulation models. The analytical models were represented by Excel, Mathematica, and MATLAB/Simulink, with some limited investigation into Satellite Tool Kit (STK) connections. The Velocity Template Language was used to render specifications of the design and analysis results from the MagicDraw data store.

In this architecture, Excel was used to house tabular data such as a master equipment list with part descriptions, quantities and masses. Mathematica was meant to handle complex computations on vectors of inputs and outputs. And MATLAB's Simulink toolbox was used to represent a dynamic simulation of power and data for the spacecraft during its science phase.

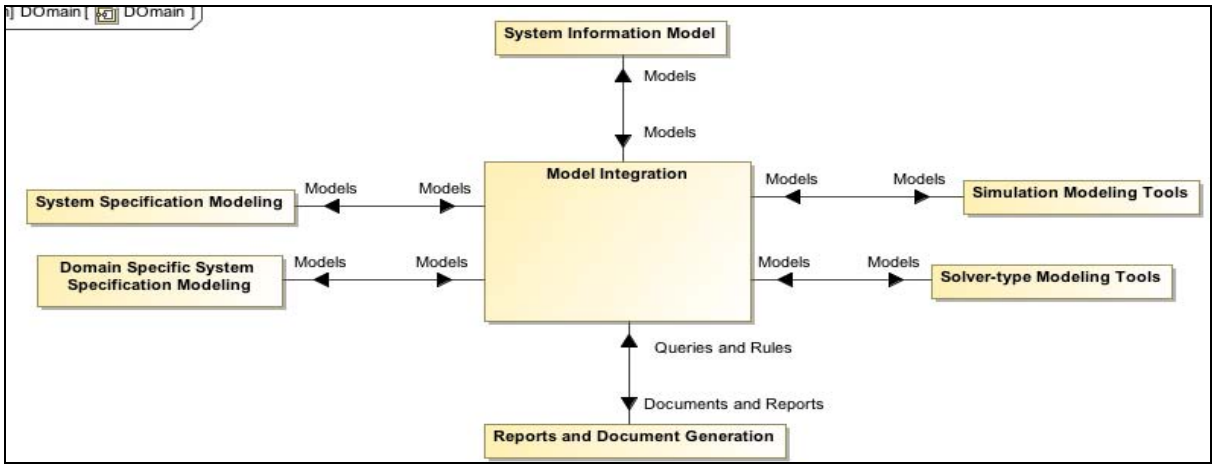

Figure 1. Concept of Integrated Modeling infrastructure rendered in a SysML IBD

Against this architecture, classes of modeling problems focused on low maturity designs were examined. The pilot has identified successes and issues with taking engineering models into an integrated modeling environment based on the quality attributes described in this section.

Prototyping Domain-Specific Diagramming In Support Of Building Information Models

Another direction of investigation for this pilot was in the area of domain-specific representation. While generic diagrams with boxes and connections can be explained to systems engineers, there are also a specific set of shapes and visual semantics that spacecraft systems engineers understand from academia or previous training/experience. In this pilot, propulsion system semantics were experimented with. In this case, the core of the information is contained within the shapes of individual icons, where connector lines have little information other than what pieces are connected together.

Two mechanisms for diagram specialization were experimented with in MagicDraw. The first is a simple overlay. The second is the ability to associate a stereotype with a new type of icon. In addition, MagicDraw affords the ability to develop custom diagrams and workspaces in order to complete the customization.

Image overlays were used for the operational domain, in which an internal block diagram (IBD) was illustrated with images of the spacecraft and bodies of interest to describe physical interactions of interest to the science community. A variety of wavelengths of light, electrical fields, charged particles, and radar reflections were illustrated in the diagram. Most of these are standard items for scientific examination, but the structure of the IBD also provides a hook to designers for interfaces the spacecraft needs to provide to its environment. Cameras must see the planetary surface, magnetic field instruments must be permeable to outside fields but shield those from the spacecraft, and so on.

Initially, the illustration approach was applied to the propulsion subsystem, but experimentation enabled the customization approach. The comparison between the two is shown in Figure 2.

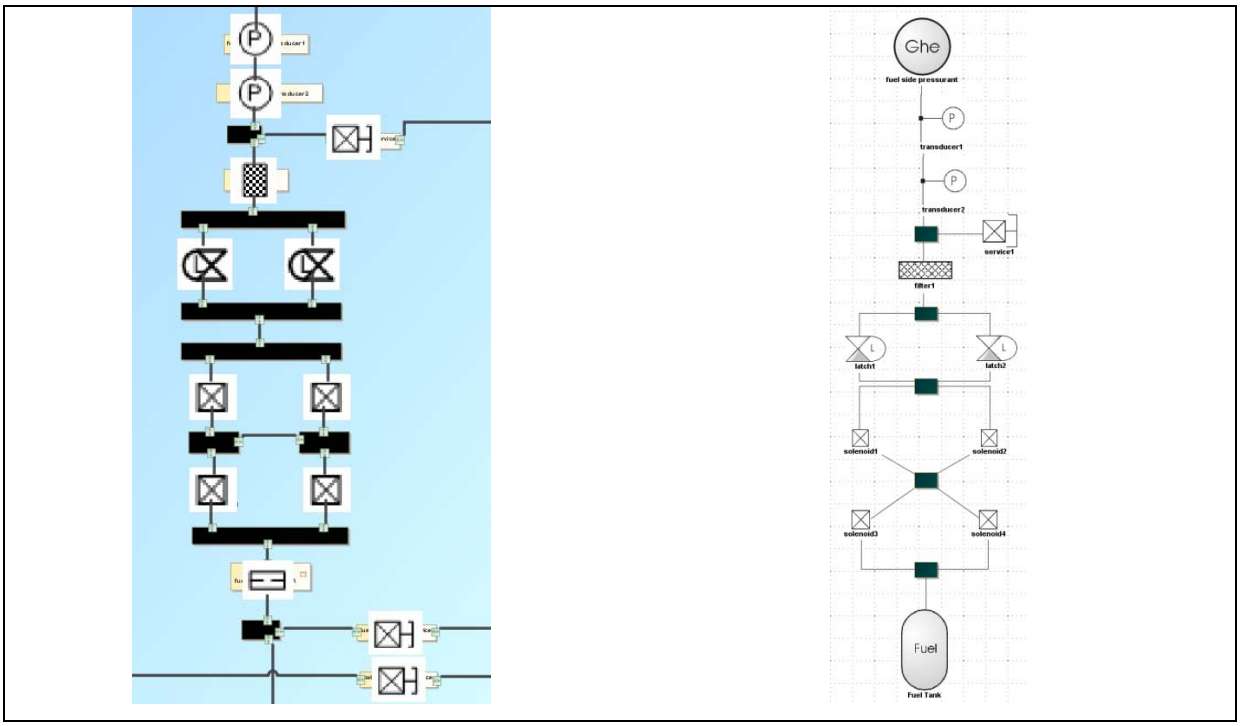

Figure 2. Diagram with image overlay vs. domain-specific diagram in MagicDraw

All property, value, and part displays were suppressed so that the icon alone and a caption were left. The result of this customization is a diagram with the visual semantics of a domain of expertise. The fact that these diagrams have already been used and found useful for a specific domain means that there is no need to convince practitioners to use the diagram. The only need is to change the platform with which the diagrams are drawn, so that the information of the domain is captured, and translatable to a basis that can be used for a variety of automated query functions.

In the above example, it is worth noting that displaying any structure whatsoever upon an instance causes the display to revert to the basic block. This is a drawback, as it precludes the diagramming of ports or values along with domain-specific information. This deficiency should be addressed at both the tool and language level.

Model Organization for Trades

An area of strength for MBSE that should be considered is conducting trade studies. In the early phases of design, trade studies are being performed not only to set requirements on subsystem parameters, but also to make major architectural decisions. Greater coverage of the design space in a given analysis provides greater assurance that the appropriate decisions have been made. With sweeps of continuous parameters, this can be straightforward to automate. Sweeping architectural alternatives requires a machine-readable specification of both the architecture and the physics that define its response to the environment.

However, much possibility is yet to be realized due to the lack of mature helper tools to generate a set of alternative architectures from candidate components and feasible connections. Although this toolset has yet to be developed, and no work was done toward it in this study, it was anticipated. The attributes of SysML that would enable rapid specification of alternatives were explored in this work.

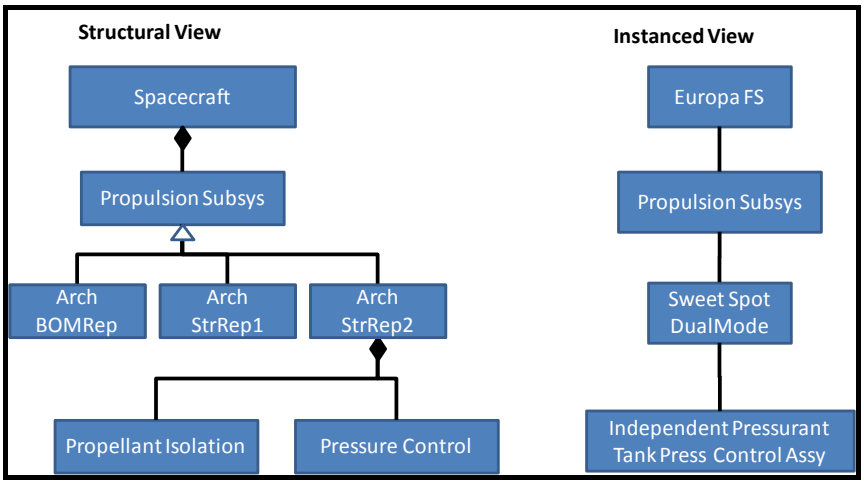

Figure 3. SysML pattern for systems hierarchy with alternate representations

The construct that was developed specifies the general structure of alternative options in a block definition diagram, as shown in Figure 3. The hierarchy of the architecture through systems, subsystems, and assemblies is shown. Also, each level of hierarchy has multiple representations – in this case, there are bill of materials (simply a parts list) and structured (how parts actually interact) versions. The SysML modeling pattern was designed to allow an automated traversal of the discrete space of options that is formed by combinations of choices at each level of the hierarchy. The information in the block definition diagram serves as a template to make system architectures from. The instance block construct of MagicDraw is

then used as a container for each architecture that is made from the template. The “instanced” architecture can then serve as a guide to customized scripts, constraints solvers, and report generation. The general scheme is shown in Figure 3.

Connecting Descriptive and Analytical Tools

The integration of analytical and descriptive tools was also investigated in this effort. Microsoft’s Excel, MathWorks’ Simulink, and Wolfram Research’s Mathematica were all attached to the SysML model in one way or another. Through the effort, each tool was viewed as a black box, and the ParaMagic plug-in to MagicDraw was used to feed inputs and retrieve outputs. Excel was used in order to load parameter values into the model en masse, Simulink to perform a dynamical analysis on a mission scenario, and Mathematica was used for calculations. Upon further reflection, it was considered that each of these tools actually has an underlying metamodel. This is not a unique realization, as Simulink blocks have been transformed into SysML model elements automatically based on the inherent semantics of Simulink diagrams (Qamar et. al, 2009). A transformation was performed in the other direction for a discrete event simulation (Huang et. al, 2007). These transformations illustrate the potential of bootstrapping descriptive models from calculation methods used by engineers in their current practice.

The Excel workbook from which parts data were loaded also had a structured meaning that could be tied to the SysML model. Each worksheet holds the information for parts belonging to a particular subsystem. Several tables were arranged on the spreadsheet, with headers indicating both assemblies within the subsystem and architectural alternatives for those assemblies. Masses and contingency allowances were held in specified columns. While a designer needs to specify these connections in the information model, they are just as rigorous as a database table definition if the spreadsheet layouts are consistent.

Mathematica has a well-structured metamodel, starting with the Expression construct. All elements of Mathematica derive from Expression, and this allows for a versatile engine that can replace one mathematical formula for another. Each element type is described by a Head, such as Symbol, List, Plus, or functions like Zeta. The engine can sort through multiple rules for replacing objects with one Head over another, and use each Head as context for which replacement rule to use. Since the structure that Mathematica uses is meant to interpret mathematical objects, it also opens the door to employing transformation methods to render expressions into desired forms in the SysML world.

Simulink diagrams have the closest relation to SysML semantics of all of the tools. Every entity in the diagram is a transfer block, source, or sink. Transfer blocks transform a set of inputs into a set of outputs. Sinks either generate starting values or pull them from outside the scope of the simulation. Sources do otherwise. Lines on the Simulink diagrams represent a direct transfer of values between blocks. Also, it is possible to encapsulate multiple blocks into a single sub-model, which provides the opportunity to map the algorithms used to analyze a given piece of hardware or software to that specific component.

However, these reflections and the examples provided of integration are very new in the context of SysML. Simulation tools typically have models of operation that are implicit at best. Extracting these models requires reflection upon the nature of the analytical models underlying them (e.g., linear, time-invariant, idealized treatment of couplings), recommended workflows, and data structures. The payoff in considering internal models of these tools is the great deal of valuable information stored within the structure of these tools.

In the pilot, black-box integration used SysML constraint blocks as a link to executable simulations. The test case for this connection was to analyze a simple sequence of propulsive maneuvers to calculate the propellant needed for a spacecraft to perform its mission. The full analysis algorithm was encoded as a Mathematica function. Parametric diagrams were used to make the connections between the system model and this analysis, as shown in Figure 4. The central block, with the constraint finalCalc: burnMassCalc, is the direct reference to the Mathematica function. ParaMagic performed acausal solving of constraint block and Value property networks with Mathematica.

ParaMagic can be contrasted with tools such as iSight and ModelCenter, which also integrate simulations. Where they are less able is the connection of these simulations to an information model of a system. Thus, the current state of tools available to an integrator requires a choice between more intuitive simulation interchange against a strong connection to an integrated systems model. As identified above, a true integration between systems and simulation models would be preferred, but a great deal of work must happen before then.

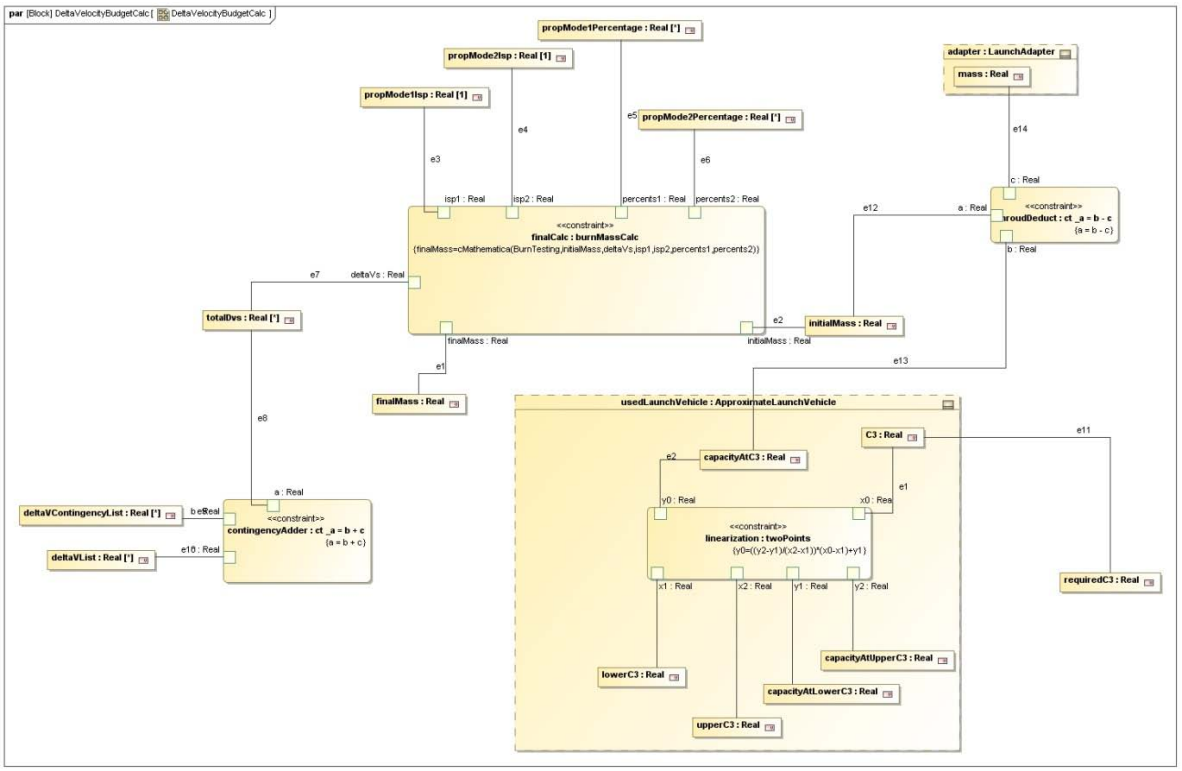

Figure 4. Parametric diagram for Europa arrival mass

Information Modeling: Usability and Communicability

SysML is a young language even though it is based on the 10+ year legacy of the Unified Model Language (UML), the basics of requirements, and the theoretical foundations of Composable Objects. While SysML brings capabilities for realizing the information modeling and diagramming roles in systems engineering, the language still lacks some fundamental systems engineering concepts. Most of those observed are based around the fact that UML and its meta-model have much in the way of software engineering concepts and principles in their architecture. In software modeling the model is a close match to the code, but in systems modeling the level of abstraction is arbitrary. The assumption that software principles enshrined in the UML meta-model are applicable to systems engineering contains

some misfits. Model construction, depiction, properties, and behavior in the SysML reflect this and thus there is a need for a true systems engineering meta-model and incorporation of things useful to systems engineers like tables and timelines. But beyond this, there are a series of constructs in the SysML language that appear at first to be compatible with systems, but miss the mark in subtle ways.

One such example observed in this pilot was that of inheritance. Since UML relies on the software concept for inheritance – a statement in the code which tells the compiler to copy or otherwise provide access to another piece of source code. This concept caused problems with inherited ports in the pilot model. The diagrams looked correct, but when generating input-output tables, Magic Draw could not differentiate ports on the parent block from ports on the child block. This was not difficult to find a work around for and it certainly is a tool issue. The crux of this issue however, is that SysML needs a meta-model for ports and interfaces that takes these systems-specific concerns into consideration. Another example is multiplicity. At one level of description, it can be quite handy to specify a simple value for the number of components within a system. When it comes time to translate this number into a structure, however, there is no way to assign separate roles to each of those components in an internal block diagram. For example, the propulsion system that was modeled contained many latch valves. To properly model the plumbing of the system, a separate Part property for each valve had to be made.

Instancing is another area that has tension between SysML's software roots and the mindset of systems engineers. In software, instances are all run-time, so their values can be dependent upon the user's input. In systems, the line between archetype and realized instance is arbitrary. The authors' view for systems is that a given instance is the thing to be built (e.g., ½" threaded bolt #121 within the system testbed) while the archetype is the design.

This leads to the issue of placing values and other content on IBD/Parametric and Activity diagrams. These are probably the single most useful set of diagrams in SysML and most natural for systems engineers. Instances on Block Definition Diagrams allow values to be captured in the model. They provide only minimal functionality (namely the ability to contain instanced values on Value properties), with no graphical utility for communicating or simplifying the design. They only serve the purpose of capturing and organizing the numerical quantities. Currently these elements are specified separately and then later unified through context specific values. This is a tedious effort to performing what amounts to assigning numbers to boxes in the eyes of the lay-modeler.

Parametrics are really behavior models and as such it is very desirable to use them as intimately with activity models as they are used with the structural aspects of SysML. For example, it makes sense to constrain a downlink activity by the results of an occultation analysis or a state transition by orbital altitude. In general, using mathematical terminology and constructs is a key aspect of behavioral specification.

Data structures and representations are another group of limitations for SysML. Systems engineers routinely use matrices, vectors, and tensors in their specifications. These concepts' absence often requires the construction of exotic object structures and stereotypes that lead to ambiguity in the model. Further, the ability to input data in a compact, structured format such as a table would allow for data entry to be performed more rapidly.

Model Manageability and Durability

The SysML language does not have a constraint on the size of the system it can describe. However, tools for managing scope are required in order to make large system descriptions

tractable. MagicDraw also lacks larger scale model management tools like the ability to see at a macro-level what has changed between model versions or how consistent the model is. There are also no tools that allow for deliberate transformation and comparison between versions of the SysML standard which would make durability more difficult to assert easily.

The above are problems for describing a single design point. As stated in other sections of this paper, formal models present great potential in increasing the breadth of searches for better design solutions. A way to compactly represent a set of combinations, from which designs can be generated, would be useful. One such structure is a Morphological Matrix. The Morphological Matrix presents a series of alternatives in a structured form, with the assumption that the choices in each row are independent (except for incompatibility between options). The multi-level version allows for the considerations of structure, as described visually in Figure 5 and utilized in software implementations (Engler, Biltgen, 2007). In the example shown in the figure, selecting a given aircraft platform speed limits the selection of options in other categories, such as type of engine that is feasible.

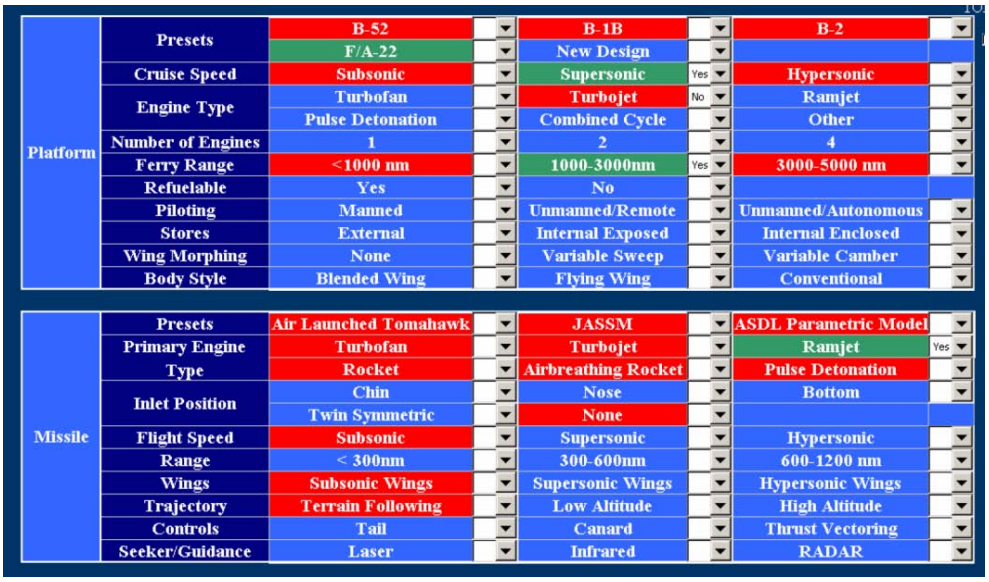

Figure 5. Representation of combinatorial trade space as an interactive matrix (Engler, Biltgen, 2007)

Derived Principles

After the pilot was completed, presentations were made to many groups within JPL. Reflection upon experiences from the pilot led to the construction of some basic principles to observe when using descriptive models for early design.

1. Model only what you need

Feedback that was given by project engineers indicates a need for the right level of modeling. Modeling excessively can lead to expectations of inaccurate formality and maturity. A good modeling heuristic is to only model what you need. An appropriate model can provide a solid foundation with the right amount of agility to still accommodate the needs of the project.

2. Draw diagrams when structures are novel or need to be audited

The fact that temporary documents such as emails and PowerPoint/Visio drawings are passed back and forth refer to the need for both written and pictorial communication. This is a moment of opportunity to capture this information for later use and elaboration.

This may be a place where a lighter weight version of tools typically used to specify a SysML model might be useful, or where the ability to easily merge files is most useful. For this particular application, the goal is for the user to sketch out the diagram and commit it to a file in a format that can be imported later if there is a desire to make the temporary information in the diagram permanent. A rigorous metamodel can also be specified in the background of this lightweight tool (or Perspective in MagicDraw) from which standardized objects can inherit.

3. Describe once, represent often

A strength of SysML drawing tools is that they store data as the user creates diagrams. A useful pattern that was discovered during this effort involves the use of the "Related Elements" command of MagicDraw. Adding a single element (block, activity, etc.) to a diagram and then using this command causes the program to generate all connections that end in that element. Judicious trimming at this point would result in a new view of interest.

Another use of automatic generation is to allow representation of a system at a desired level of detail. If there are multiple levels of hierarchy in a system, then automated generation can call up one level after another until the new diagram is complete. This simply requires that each level of the hierarchy is represented by a Part property that refers to a block with its own internal definition. This was applied to the propulsion system as shown in Figure 6.

At first, the value in this exercise may seem only to be redrawing a few diagrams. However, this effort resulted in a model of sufficient complexity for members of the team to forget what elements were connected to each other. The use of automated generation in this case served as a way to audit the model, since all of the diagrams that had been drawn only involved a few elements in order to maintain simplicity. Automatically generating the system model on the infinite canvas of the computer screen can also allow systems engineers to audit its maturity and completeness.

Also, as mentioned before, analytical and support tools should be drafted into this principle. The design information input into an analysis tool should be recoverable. More than that, the synergy of parameter inputs and the underlying information structure of the analysis should move toward refinement of the system design.

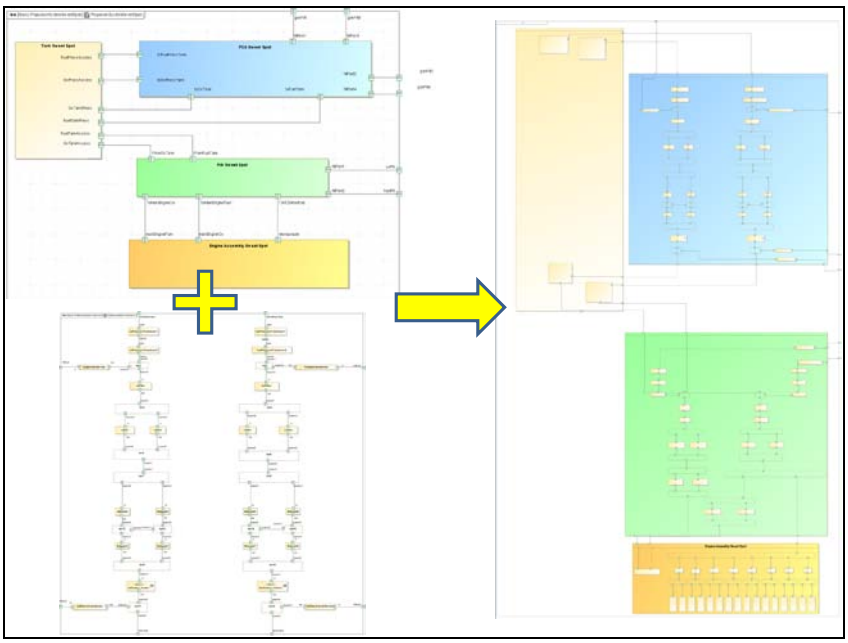

Figure 6. Automated generation of integrated system diagram

4. SysML is a language – start new users with elements similar to their own language

During its introduction at an organization, the work of developing a model will likely emulate that of defining requirements. It is very difficult to make a clear English sentence that all readers will interpret the same way. It is equally difficult to make a machine-readable statement that captures the original intent of the modeler. Modeling will likely involve interviews with subject matter experts. If the "playback" representation is chosen carefully, it has been found to greatly speed up the process of convergence of understanding between modeler and subject expert. But, it is important not to interpret the inclusion of systems engineers as a need to "dumb down" the back-end infrastructure of the model. A well-built model is still necessary to enable efficient and useful queries and transformations of the model downstream.

Representations of the design should either use familiar symbol semantics or intuitive ones. Many questions of "what does the black diamond mean?" came up during discussions in the pilot. As described below in the table, various SysML diagrams have different levels of connection to the usual concerns of system engineers. For example, the internal block diagram is based in the semantics of connections and interfaces between the constituents of a system (or subsystem). In order to bring SysML to systems engineers, the most natural semantics should be emphasized and introduced first.

One thing that became clear through this trial effort and others in the systems engineering group at JPL is that a select number of SysML diagram types were found to be perplexing or foreign. The most oft-cited diagram that garnered this reaction was the block definition diagram, which is a systems-flavored descendant of the UML class diagram. At least two of the major texts on SysML (Friedenthal, 2008; Holt, Perry, 2007) present this diagram type first, showing how the software meta-model and practitioner mindset of UML is in tension with SysML's focus on systems engineers.

More appropriate diagrams for introducing SysML would be the activity or internal block diagrams, which have a graphical syntax (in their most basic forms) that is akin to what is already in use. These diagram types are also appropriate for early design. Activity diagrams can be used to sketch out concepts of operations, mission designs, and science collection schemes. Internal block diagrams are appropriate for representing schematic "cartoons" of computer boards and data routing or identifying what parts of hardware will be organized under what subsystem.

Table 1. Attributes of diagrams for early design

Diagram Type	Applicability to Early Phase	Familiarity
Activity	Describes high-level operational concept and identifies critical events	Basic version very similar to notional diagrams drawn to show flow of events
State Diagrams	May be used to highlight states modes and phases of the mission	Varies on experience and background.
Block Definition	Used to describe definitions and instances of parts to be used in subsystems.	Little
Internal Block	Define alternative architectures for subsystems (e.g., monopropellant versus dual mode propulsion system)	Similar to notional block diagrams, ports easy to understand, some semantics new to project engineers

5. Leverage diagrams for execution

Early phases of design are oriented toward analysis and identification of suitable levels of performance for the spacecraft and its various subsystems. The design is a moving target, and so at this phase, there is not much reason to describe it in detail. However, when the frenetic burst of early design activity is over, a description of both the design and its rationale will need to be produced. There is the rub.

Automatically retaining descriptions of design alternatives allows for the construction of a richer record of support for design decisions. This is a valuable result of a model-based approach, but it may be insufficient to repay the efforts invested in training users and developing models. The value of these artifacts may be increased if they are developed not as simple descriptions, but part of the core data structure behind analyses.

A very simple version of this was done with a launch date sensitivity analysis that was conducted during the effort. In it, a series of candidate launch vehicles were modeled, as were alternate trajectories. These were used to make a full combinatorial matrix to see how much launch mass could arrive at destination for a given launch window with a given launch vehicle. A script automatically reported this matrix, and although the combinations were built by hand, a script could also construct the combinatorial space of options. Each alternative was created with instances of Blocks that described the launch vehicles and the trajectories, and a Mathematica notebook was used with the ParaMagic plug-in to calculate the delivered mass of each of the alternatives.

6. Understand simulation tool meta-models

Simulation models are core to engineering design. If these models are well-structured, they can provide the properties needed for integration with the information model as well as with other simulations and tools. This will ease integration efforts but may require additional work for successful execution. For example, the simulation of the instruments in the pilot would have to be broken up into separate simulations for each instrument since these interfaces are internal to the simulation and cannot be explicitly accessed through Matlab or ParaMagic.

7. Don’t pollute your system model

The information model provided by SysML is not a general purpose database. It is not intended to store thousands of alternatives, trade space explosions and records of various activities. It is meant to provide architecture and specifications. Two to three key detailed alternatives are appropriate, but it will not support huge numerical crunching activities that are better suited to other tools. A systems model is best suited to capturing the essence of the result of those activities and to generating data for them. In the larger picture, SysML tools should aim to integrate with other tools that support these kinds of problems more robustly.

8. Keep systems engineers involved and work on real problems.

Engineers used to working with a particular paradigm present apprehension about losing capabilities and interfaces they are familiar with and/or that are required to achieve the objective of their engineering task. This is a valid concern, and developers of the model-based paradigm need to work closely with users to speed adoption. Also, as this pilot has made clear, useful work toward model-centric environments is becoming less conceptual and more focused on implementation. This is also a sentiment encountered while trying to advance MBSE at JPL – the precepts of MBSE are lauded as important, but skepticism focuses on the maturity of implementation approaches. Keeping systems engineers involved keeps the problems of implementation in focus.

A good example of this is in the limited data representations that are allowable in SysML. Systems engineers like designing with diagrams, however they equally prefer a tabular interface. Until the SysML adds tables or CASE tools start supporting editable tables, this interface must be added separately to ensure engineers have what they need to do their work.

Summary

Overall the pilot was successful in demonstrating the capabilities and some advantages and challenges of MBSE and integrated modeling over the document based paradigm. The IMCE pilot worked with currently existing model specification tools and simulations to understand what happens when rubber meets road. Issues were found with the SysML language, the current disposition of simulation tools, and the way that information modeling is presented to systems engineers. However, the pilot also found that tools for specifying systems, such as SysML, are mostly ready to do their jobs. Further, efforts on SysML/simulation tools such as ParaMagic and the scripting capabilities of both the simulation tools and MagicDraw are also both well on their way to success.

Acknowledgements

This work was funded under the Integrated Model-Centric Engineering initiative at JPL. The authors are thankful for financial support on this work through that initiative. Also, the authors would like to thank the Outer Planets Flagship systems team for the generous use of their time, as well as their expertise and viewpoint. Their candid and forthright critiques of the work done on this pilot are already being applied to updated approaches and new lists of problems to tackle on moving model-centric engineering forward.

Biographies

Bjorn Cole *is a new career hire in the Misison Systems Concepts section at the Jet Propulsion Laboratory. His research interests include systems design, optimization, simulation, and design space exploration and visualization. He earned his B.S. in aeronautics and astronautics from the U of Washington and M.S. and Ph.D. degrees from Georgia Tech in aerospace engineering.*

Christopher Delp *is Architectural Engineer for Multi-Mission Operations and a member of of the Flight Software Systems Engineering and Architectures Group at the Jet Propulsion Laboratory. His research interests include software and systems architecture, model-based systems engineering applications, and real-time embedded software engineering. He earned his M.S. and B.S. degrees from the U of A in Systems Engineering.*

Kenny Donahue *is a summer student in the Mission Systems Concepts section at the Jet Propulsion Laboratory. He is finishing up his Masters at MIT, focusing his research in passive recognition of objects in lidar point clouds for use on an autonomous forklift. His interests are varied but include automatic speech recognition, natural language processing, and system design. He earned his B.S. in Electrical Engineering/Computer Science from MIT.*

References

Delp, C. et. al., INCOSE Model Based Systems Engineering Challenge. Contribution of the Space Systems Working Group. V3.0, 2009.

Engler, W.O., Biltgen, P.T., Mavris, D.N., Concept Selection Using an Interactive Reconfigurable Matrix of Alternatives (IRMA). 45th AIAA Aerospace Sciences Meeting and Exhibit, 8 - 11 January 2007, Reno, Nevada.

Estefan, J., Survey of Model-Based Systems Engineering (MBSE) Methodologies. INCOSE-TD-2007-003-02, 2008.

Friedenthal, S., Moore, A., Steiner, R., *A Practical Guide to SysML: The Systems Modeling Language*. Morgan Kauffman, San Francisco, CA, 2008.

Herman, I., Introduction to the Semantic Web. W3C Presentation. Accessed online at [http://www.w3.org/2009/Talks/0829-Nanjing-IH/] on October 30, 2009.

Holt, J., Perry, S. *SysML for Systems Engineering*. Professional Applications of Computing Series 7, Institution of Engineering and Technology Press, Herts, UK, 2007.

Huang, E., Ramamurthy, R., McGinnis, L. F., System and simulation modeling using SysML. *Proceedings of the 2007 Winter Simulation Conference*.

Ingham, M., Rasmussen, R. Generating Requirements for Embedded Systems Using State Analysis. IAC-04-IAF-U.3.A.05

Kordon M., Wall, S. Model-Based Engineering Design Pilots at JPL.

National Aeronautics and Space Administration. *Jupiter Europa Orbiter Mission Study 2008: Final Report.* NASA, 2008.

———. *NASA Systems Engineering Processes and Requirements*, NASA Procedural Requirement, NPR-7123.1A.

Object Management Group, *OMG SysML v1.1*, www.omgsysml.org

Peak, R.S., Burkhart, R.M., Friedenthal, S.A., Wilson, M.W., Bajaj, M., Kim, I., Simulation-Based Design Using SysML—Part 1: A Parametrics Primer. INCOSE Intl. Symposium, San Diego, 2007

Qamar, A., During, C., Wikander, J., Designing Mechatronic Systems, a Model-based Perspective, an Attempt to Achieve SysML-Matlab/Simulink Model Integration. 2009 IEEE/ASME International conference on Advanced Intelligent Mechatronics, Singapore, July 14-17, 2009.

Weilkins, T., *Systems Engineering with SysML/UML*. Morgan Kauffman, San Francisco, CA, 2008.

Wymore, A. W., *Model Based Systems Engineering*, CRC Press, 1993