Open access • Proceedings Article • DOI:10.1109/AERO.2013.6496926

# Model based document and report generation for systems engineering



Christopher L. Delp, Doris Lam, Elyse Fosse, Cin-Young Lee

**Institutions:** California Institute of Technology

**Published on:** 02 Mar 2013 - IEEE Aerospace Conference

**Topics:** View model, Model-based systems engineering, Systems Modeling Language and Process (engineering)

# Model Based Document and Report Generation for Systems Engineering

**Christopher Delp, Doris Lam, Elyse Fosse, Cin-Young Lee**
**Jet Propulsion Laboratory, California Institute of Technology**
**4800 Oak Grove Drive**
**Pasadena, CA 91109**
**(818)319-3251**
**Christopher.L.Delp@jpl.nasa.gov**

***Abstract*—As Model Based Systems Engineering (MBSE) practices gain adoption, various approaches have been developed in order to simplify and automate the process of generating documents from models. Essentially, all of these techniques can be unified around the concept of producing different views of the model according to the needs of the intended audience. In this paper, we will describe a technique developed at JPL of applying SysML Viewpoints and Views to generate documents and reports. An architecture of model-based view and document generation will be presented, and the necessary extensions to SysML with associated rationale will be explained. A survey of examples will highlight a variety of views that can be generated, and will provide some insight into how collaboration and integration is enabled. We will also describe the basic architecture for the enterprise applications that support this approach.**

## TABLE OF CONTENTS



## 1. MBSE AND THE STATE OF THE PRACTICE OF DOCUMENT GENERATION

Several projects at JPL have now embraced Model Based Systems Engineering (MBSE). As a result, JPL has developed an institutional approach to MBSE. This approach is based on Systems Modeling Language (SysML) [1] and formal ontology expressed in the terminology and lexicon of each engineering domain. MBSE promises to alleviate the difficulty systems engineers face in communicating across engineering disciplines primarily in terms of completeness and consistency. By describing these systems in a formal way using domain specific terms, models can be checked for completeness and consistency. These models can also be analyzed to answer questions about the system such as input to simulations or other engineering analysis.

At the core of realizing these benefits is effective communication between Systems Engineers and other engineering disciplines. Since other engineering disciplines are not versed in Systems Engineering models, Systems Engineers still need to produce documents and reports as the primary way to communicate with stakeholders and other engineering disciplines. One of the keys to MBSE adoption at JPL has been the practice of generating documents from systems engineering models. This allows systems engineers to easily update and ensure consistency among a set of documents as updates are made to the model.

This document generation technique originated from other JPL efforts including Ops Revitalization [2]. Since these initial innovations, MBSE at JPL has flourished in a number of projects. In particular, the Ops Revitalization Task [3], the Europa Study [4] and the Integrated Model-Centric Engineering effort [5] have been crucial drivers for the development of models, architecture, technology, and applications that provide this capability.

As MBSE practice has begun to move into the mainstream, several homegrown approaches have been developed around the use of the DocBook standard for publishing [6]. In general, these approaches involve the use of a SysML profile for DocBook to produce a model of a document. The document model is then linked to other SysML models and diagrams to produce the document.

These approaches are effective at generating the basic structure of the document with injected model information. However, they lack the semantics and patterns to describe how the model is projected into a document structure. Each existing implementation has attempted different ways to support this, but none of these applications provides a comprehensive set of capability. They also lack a more fundamental concept and foundational support for describing how to extract information from the model in such a way so that analysis and editing of that information can be integrated with external applications.

MGSS Ops Revitalization [7] and the Europa Mission Study [8] have deployed full-scale project models in SysML. Several other efforts across industry are engaged in MBSE with a similar scale of modeling effort [9]. Modeling at an enterprise scale requires enterprise computing environment capable of supporting collaboration among a variety of users working with large models and data sets. Web technologies have been used extensively in these efforts to realize such a scalable enterprise computing environment.

This paper describes the fundamental concept of Viewpoint and View as the foundation for providing a comprehensive capability for generating Views of models. The architecture for Viewpoint and View and its extensions in SysML are

978-1-4673-1813-6/13/$31.00 ©2013 IEEE.
[1] IEEEAC Paper #2233, Version 2, Updated 5/1/2013.

described using examples from the projects at JPL sponsoring this work. Models of this size require enterprise scalability. Finally we describe the current implementation of a Model Based Engineering Environment and the document generation support and applications for generating documents and reports for Systems Engineering.

## 2. The Principle of Communication

Systems Engineers and Architects produce products that must communicate with a diverse group of customers including different engineering disciplines, managers, organizational and business roles. This diverse group each has a different point of view with respect to how they understand the system. This motivates a principle of communication that will ensure that the system is described from each the point of view of each of these stakeholders. 2-way communication asserts that person communicating report what they heard the other person request as well as there response. The ISO/IEC 42010 [10] definition of Viewpoint and View is consistent with this principle. Viewpoint and View can be used to provide a platform that can describe different aspects of a model according to the rules for describing those different aspects of the model. Viewpoint describes what the stakeholder point of view and View represents the depiction of the model of the system according to the Viewpoint.

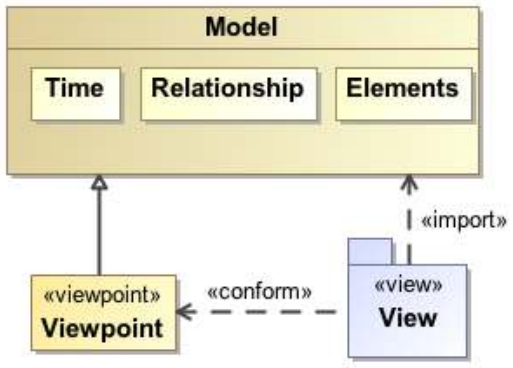


**Figure 1**. Metamodel of Basic Viewpoint and View

Generating documents and reports using Viewpoints and Views has been demonstrated at JPL as an effective way to communicate across disciplines using models to ensure completeness and consistency of the system architecture and design. The current technique employed at JPL uses SysML Viewpoint and View to specify a model for communicating different aspects of a system model. The SysML definitions of Viewpoint and View are consistent with ISO/IEC 42010. Figure 1 illustrates the basic semantics for relating elements of the model to the model View. The conformance relationship expresses the requirement that the View of the model be consistent with the methods and rules expressed by the Viewpoint. It is often necessary to communicate a certain set of Views in a particular order. These collections can be represented as familiar document structures such as sections and subsections in a document as well as slides, tables, worksheets or other forms typical in office reporting software.

The semantics of Viewpoint and View are represented mathematically by stating that a Viewpoint morphs the elements of a model into contents of the View as seen in Figure 2.

If VP is defined to be the homomorphism that represents a viewpoint then:

$$\text{VP} : D(\text{VP}) \rightarrow R(\text{VP})$$

where D(VP) is the set of integrated model elements that are within scope for the Viewpoint (e.g., the domain of the Viewpoint) and R(VP) is set of view elements that is the image of D(VP). (e.g., the range of the Viewpoint). It follows then that:

$$\text{View} : \{\text{VP}(\text{ME}) : \text{ME in } D(\text{VP})\}$$

where ME corresponds to a model element. In other words, a Viewpoint is the homomorphism that transforms a subset of model elements into View elements.

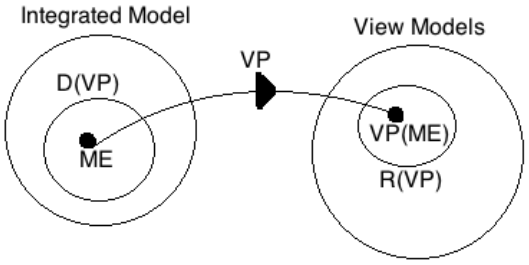


**Figure 2**. Mathematical representation of Viewpoint and View

Representing Viewpoint and View mathematically provides a theoretical foundation for the semantics - the implication being that the mathematical theory provides constraints for the implementation.

## 3. Architecture for Extending SysML Viewpoint and View

Using the Viewpoint and View definitions in SysML it is possible to define a model of Views that will provide a linearized description of models referenced by the Views. SysML Viewpoint and View have roots in ISO/IEC 42010 so most of the elements in SysML come directly from the ISO/IEC 42010 meta-model. The current SysML implementation does not treat all of these elements as first-class model elements. Table 1 identifies the concepts in SysML related to Viewpoint and how they are expanded to facilitate View generation.

*Models, Views and Viewpoints*

Most MBSE practitioners at JPL link their Views together to linearize a particular description of a model or models. Modeling the relationships between Views in this way allows for a clickable navigation through the model as well as provides a structure that can be used to generate documents and other formatted output based on the content of the model.

Figure 3 illustrates how Views can be linked together with dependencies to model the precedence order for reading the Views. Views import models of any sort or type. These models may be SysML models, ontologies, structured data from a database or website, and notional illustrations, just to name a few. In principle, the Viewpoint is even capable of describing Views that exist outside of software, such as renderings from a 3D printer or clay models of a concept automobile or building.

**Table 1. Extensions to SysML**

| SysML Element | Metaclass | Metaclass Change | Description |
|---|---|---|---|
| Viewpoint (Existing) | Class | No Change | The element that embodies the rules for describing a view |
| View (Existing) | Package | No Change | The element representing the View produced from the model |
| Conforms (Existing) | Dependency | No Change | Represents the relationship between the View and the Viewpoint that the View is required to conform to. |
| Import (Existing) | Dependency | No Change | Links the model(s) to the Viewpoint through the View |
| Stakeholder (Existing) | Tag Value (String) | Actor | The elements that represent stakeholders for the View |
| Concern (Existing) | Tag Value (String) | Tag Value or Class | A subject of interest being addressed by the View |
| Purpose (Existing) | Tag Value (String) | No Change | A narrative description of the purpose of the Viewpoint |
| Method (Existing) | Tag Value (String) | Activity Class | Behavior model that defines the ordered steps to making the View |
| Analysis Model (New) | N/A | Constraint Property | The individual analysis definitions used by the Viewpoint Method |
| View Format (New) | N/A | Property | The rules for outputting the View in specified formats |
| View Presentation (New) | N/A | Property | The styles used to present the View |
| Imported Model (New) | Tag Value from Conform Dependency | Reference Property | The parameter that is assigned the list of models described by the import |
| Model Language (Existing) | Tag Value (String) | Property | The Modeling language(s) used in the imported model. |

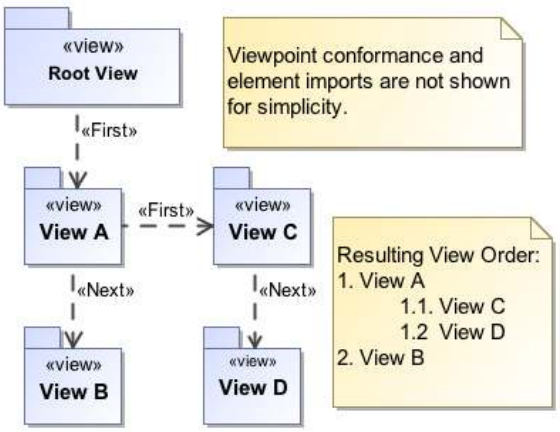


**Figure 3**. View Tree

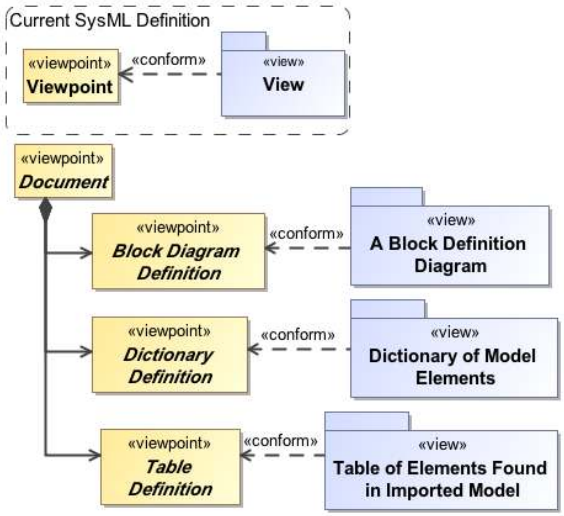


**Figure 4**. Viewpoint Templates

As illustrated in Figure 4, the Viewpoints can be composed to create a template for a particular set of Views in a particular order. This has the effect of instantiating the Viewpoint tree. It also allows a particular View tree to be compared for conformance to the Viewpoint tree.

For example, Ops Revitalization is building a series of documents that describe processes for different engineering disciplines in mission operations. The precedence and Views are the same for each discipline. The only variables are the process models. Figure 5 illustrates an example of 2 different View models that use composite Viewpoints to assert the same precedence order.

*Viewpoint and View*

A Viewpoint is a specification of the conventions and rules for constructing and using a View for the purpose of addressing a set of stakeholder concerns. The Viewpoint model as

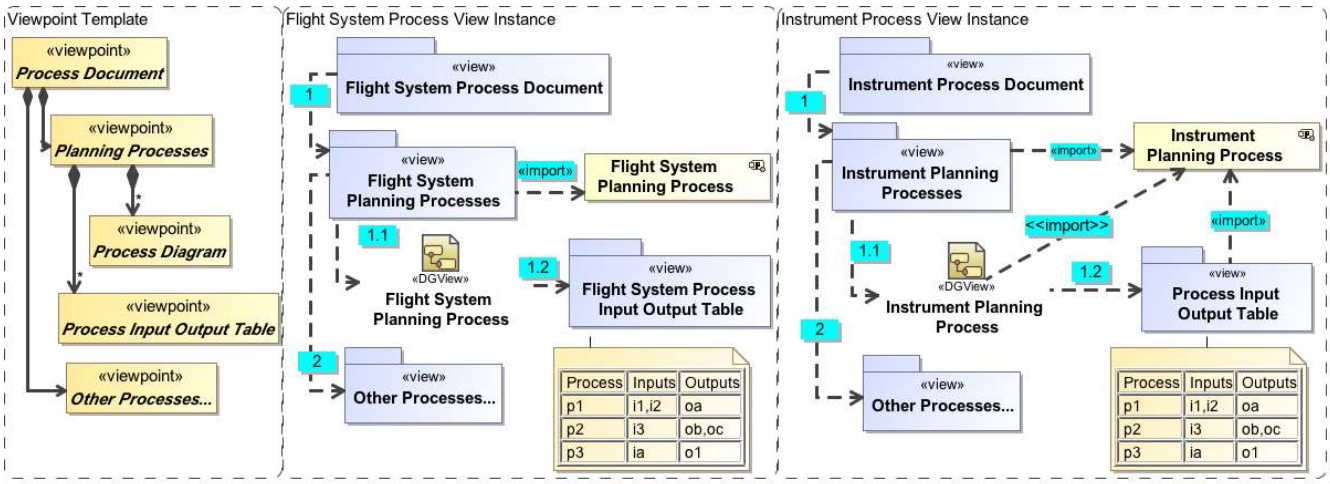


**Figure 5**. Ops Revitalization Process Documents

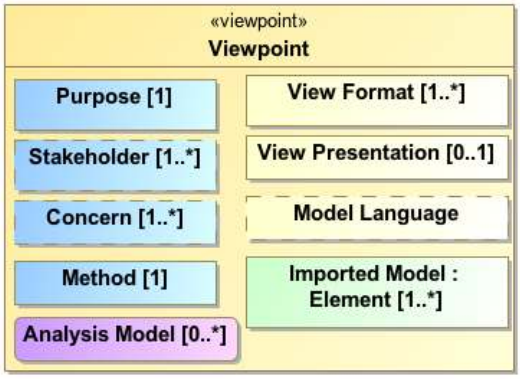


**Figure 6**. Viewpoint Model

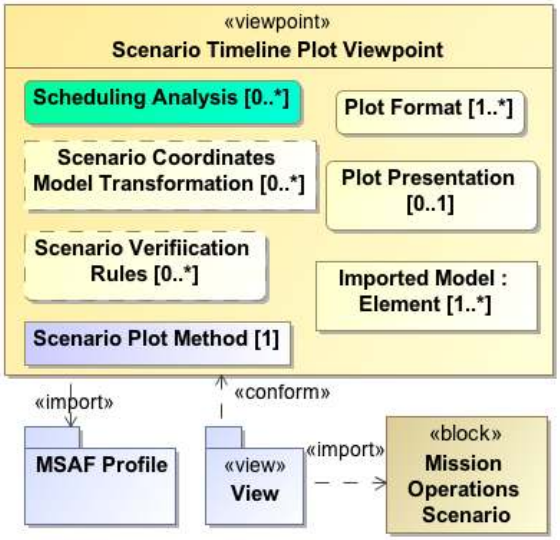


**Figure 7**. Scenario Timeline Plot Viewpoint

illustrated in Figure 6 defines the properties and constraints used to define the View. The Viewpoint also defines the Method, which is the process for constructing the View.

The Purpose, Concern and Stakeholder elements are properties that describe the point of view of the stakeholder. The Method describes the systematic process in which the model will be used to create the View. The Imported Models represent the models that the Viewpoint operates on for a given View. The View in these models is just a proxy for attaching properties and relationships. Execution of the method is necessary to render the View.

An example from Ops Revitalization is illustrated in Figure 7. This Viewpoint is defined to render a 2 dimensional Cartesian plot of an Ops Scenario model, such that the scenario function calls are plotted against time. The Ops Scenario Model is a SysML sequence model with domain specific semantics from The Mission Service Architecture Framework (MSAF). In this illustration the scenario models as well as all of the languages that are used in rendering the View are shown. The Viewpoint is defined in terms of the analyses, method, format and presentation necessary to produce the View. These elements are defined in Table 2.

An example from the Europa Mission Study [11] is the Mass Properties Viewpoint as illustrated in Figure 8 and Table 3. The purpose of this Viewpoint is to calculate the dry mass of the Flight System and show a table of components and their masses. Operating on the composite model of the Fight System through the Viewpoint renders this table. This model describes the complete component composition of the Flight System as well as the value and behavioral properties of the system.

*Domain Specific Models and Languages*

A key piece of effectively communicating with Views is specifying the language the model is written in. Modeling languages provide the patterns and syntax used in the description of the View. Domain Specific Modeling Languages specify the elements expected to be represented in the View, and may be formally or informally defined. Views are descriptions intended to communicate, thus it is necessary to assert the allowable syntax and syntactic environments that can be used to describe them. For Viewpoint the Language specified is allowed to be anything from natural language English to SysML to a Domain Specific Modeling Language to a formal Mathematical notation such as MathML. Unless explicitly

**Table 2. Scenario Viewpoint Elements**

| Viewpoint Element | Description |
|---|---|
| Scheduling Analysis | This analysis reasons out the temporal ordering from the model |
| Scenario Coordinates Model Transformation | Transformation from SysML Scenario model to trajectories in 2D coordinates |
| Scenario Verification Rules | Completeness and Correctness rules for verifying |
| Scenario Plot Method | The order for executing each analysis that ultimately produces the View |

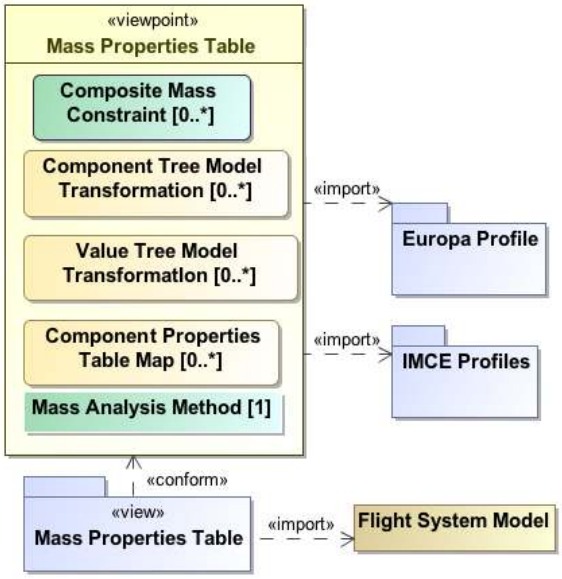


**Figure 8**. Mass Properties Table Viewpoint

prohibited, natural language documentation and narration are always expected to be included.

An example of a Domain Specific Modeling Language can be found in the Ops Revitalization project at JPL. Ops Revitalization has developed the Mission Service Architecture Framework (MSAF) [3] for the purposes of modeling Mission Operations Systems. The MSAF is a set of modeling elements and relationships for describing the interfaces, functions and process that make up an MOS using the lexicon of Mission Operations. The MSAF also defines patterns that reflect the allowable combinations of these domain specific terms. The MSAF is a Domain Specific Modeling Language and as such is built as an extension to BPMN (Business Process Modeling Notation) and SysML. Viewpoints defined for the Ops Revitalization Task are typically specified in the language of the MSAF, however sometimes SysML or BPMN are used.

*Model Analysis*

In order to generate a View of a model, it is necessary to analyze the model. The Viewpoint also defines a set of analysis that can be specified. These rules provide the means to check and/or operate on the model as part of creating the View. This

**Table 3. Mass Properties Viewpoint Elements**

| Viewpoint Element | Description |
|---|---|
| Composite Mass Constraint | The constraint that asserts that the mass of a component is the sum of the masses of its child components |
| Component Tree Model Transformation | Model Transformation that transforms the SysML flight system model into a tree of flight system components |
| Value Tree Model Transformation | The model transformation that transforms the flight system model into a tree of mass properties |
| Component Properties Table Map | The model transformation that transforms the flight system model into a map of the trees described above |
| Mass Analysis Method | The ordered steps for performing the mass analysis |

property can be used to describe any kind of analysis to be performed on the Model. Some common uses include model querying and filtering, asserting model verification, asserting mathematical formulae, and model transformations. These examples illustrate the broad range of the types of analysis that can be defined as part of the Viewpoint. It is important to note that the Method property described later defines how these different suites of rules may be applied in the course of generating the View.

The Europa study has found utility in this aspect of the Viewpoint [12]. The Viewpoints for the Flight System Mass Equipment List (MEL) define tables that describe the mass needs and constraints for the Mission. Using the model of a candidate Flight System, these Viewpoints are used to render a View of the Flight System in terms of the MEL. The Viewpoint defines analysis for verifying the correctness of the model, verifying the mass calculation, and transforming the model into a simpler model of hierarchical components and mass properties.

Transforming the model into a simpler model of hierarchical components and mass properties is an example of an Analysis that performs a Model Transformation. The Europa Flight System Model is built in SysML. It has a hierarchical component structure decorated with many properties and behaviors. In order to calculate the mass of a Flight System, the Flight System Model is transformed into a simpler model that consists of components, mass constraints, and mass properties. This new structure can then be used to solve the mass constraints and calculate the mass of the Flight System as defined in the Mass Calculation analysis.

Another analysis example from Ops Revitalization involves pattern analysis. The MSAF mentioned earlier describes the fundamental architectural patterns for a Mission Operations System. Viewpoints defined for the MSAF all include rules that verify usages of the framework patterns. These rules compare models that have been built using the MSAF and identify conditions that are not consistent or complete with respect to the pattern.

*View Format and Presentation*

Stakeholders may have conventions, organizational or institutional practices, and standards that influence how the View is to be rendered. Views of the system model that are created by Systems Engineers usually have very customized styles and presentation requirements. Different organizations may additionally prefer a variety of formats. Some views are generated in Power Point slides others are tables, documents, HTML web pages, or 3D CAD Generated animations. Additionally, conventions may dictate the use of certain diagrams, tables color codes, etc.

Utilizing these rules is key to communication. The Format and Presentation properties can be used to capture the specific rules for the View as part of the overall Viewpoint specification. The Format and Presentation properties of Viewpoint provide the means of describing the styles in which the View is presented and the output formats. Different Views require different formats and presentation styles depending on the stakeholder and the information being communicated. The examples that describe this are best discussed as part of the Method.

*Method*

The method is probably the most significant expansion of this approach. The Method is the behavior model of the Viewpoint. It describes the ordered steps required to process the model and render the View of the model according to the properties of the Viewpoint. This includes when and where to execute the analysis specified by the Viewpoint and how to apply the format and presentation specifications. The Method is also extensible to any other step necessary to generate the View.

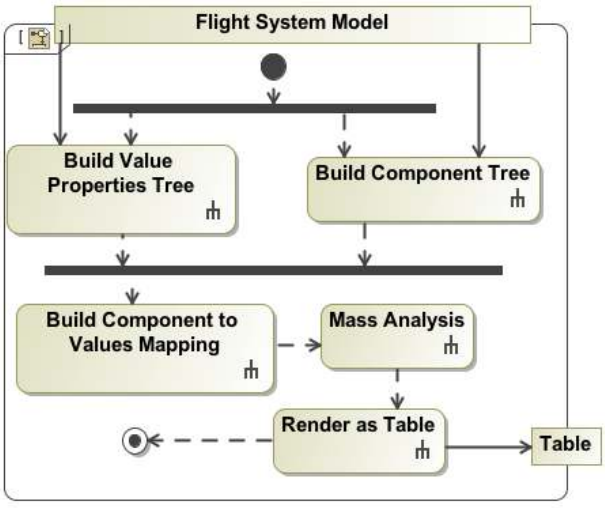


**Figure 9**. MEL Viewpoint

For example, the Method for the Europa Mission Study MEL Viewpoint is illustrated in Figure 9. It describes the steps of expressing a SysML model of Flight System components and properties as a table. This is accomplished by using model transformations to build a tree of components and a tree of mass properties and a map that relates each set of mass properties to the corresponding component. These transformations abstract out all the parts of the flight system model that have nothing to do with the Mass Analysis. The Mass Analysis asserts the constraint that the Mass of a component is the sum of the components that compose the component. This model is then transformed into a table model. Once in the table model, the format and presentation rules are applied. In this example, these tables have a long list of applied formats and presentations. For reporting, the DocBook format is used to produce a static output of the table in HTML and PDF. The Viewpoint also defines rules for rendering the table in an editable format for web browsers and Java applications. In this rule the mass values and component names are editable so that they can be easily updated without having to open the thick model editor just to change certain parameters in a lightweight fashion.

Similarly, Figure 10 from Ops Revitalization shows the Method for transforming a scenario Model expressed using SysML sequence diagrams as a plot of events and states over time. First the rules for a complete and correct scenario model are executed. Then a model transformation is used to transform the SysML Sequence model into a precedence ordered table of events. Then an analysis is performed to determine the explicit and relative times for each event within the table. Finally the plot is produced according to the format and presentation rules. The plot is currently produced in Excel spreadsheets, but ideally the Viewpoint will be able to utilize more robust tools such as Mathematica, Matlab, and Maple.

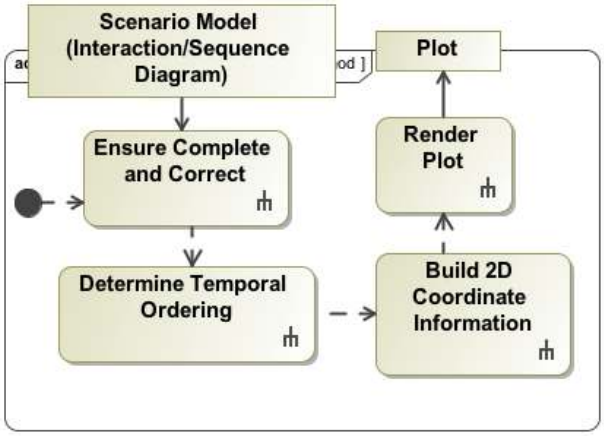


**Figure 10**. Scenario Viewpoint

## 4. Model Based Engineering Environment

For any non-trivial system to be successfully engineered, significant collaboration is required amongst Systems Engineers, Domain Engineers, Project Managers, and other related stakeholders. Views and Viewpoints form the foundation for collaboration in a model-based engineering environment as they describe how to communicate relevant aspects of the system to particular stakeholders. While the Views generated from Viewpoints can take the form of familiar documents (e.g., Interface Control Documents, Software Requirements Documents, etc.), a Viewpoint method can just as easily describe how to generate editable web views or Mathematica notebooks. As one can imagine, these dynamic views are a much more effective means for collaboration between engineers than static documents.

No tools currently support the vision of Views and Viewpoints as the cornerstone for facilitating collaboration and

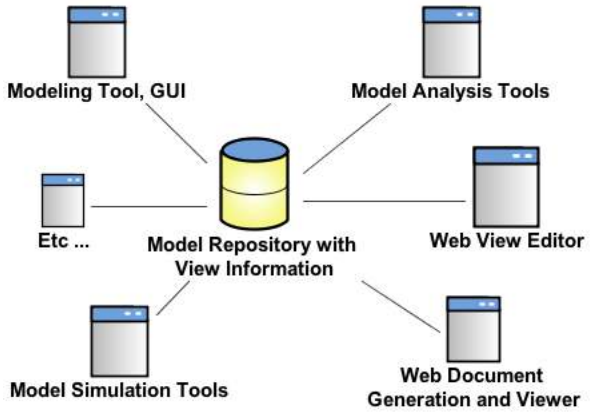


**Figure 11**. Model Based Engineering Environment

communication between systems and domain engineers for model based engineering. Figure 11 illustrates the Model Based Engineering Environment or MBEE, that is currently being developed by the Operations Revitalization and Europa Mission tasks. The MBEE consists of a model repository that serves as the single source of truth of for system models. The repository exposes all the model elements on the web via RESTful (REpresentation State Transfer) APIs. Any client, be it a SysML modeling tool, Mathematica, or whatever else, can then easily retrieve and update model information based on said APIs. This approach parallels the View/Viewpoint architecture, as the repository provides the model data, clients have viewpoints of interest, e.g., a Mathematica power usage viewpoint, which the client can then use to query out an appropriate view, say for a particular flight system. The choice of a RESTful architecture enables the enterprise scalability necessary for the largest and most complex projects.

As with other web technologies, mashups of client services can be orchestrated and combined to achieve more sophisticated analysis and simulation than any single client by itself can accomplish; for example, results of power simulations can be used to inform thermal simulations.

The capabilities provided by this environment allow systems engineers and modelers to build the model using commercial SysML tools and also allows domain engineers to input their data using more domain specific Views. For example, using the same techniques of View generation from Viewpoints, we can generate table Views of the model, which can then be edited online or used for analysis with Mathematica, Excel, NX, Maple, etc. and the results of such analysis can be fed back into the model as necessary.

This interplay between systems and domain engineers needs to be a managed and repeatable process. As the tooling and software infrastructure for MBEE has been developed at JPL, multiple projects have converged on the process shown in Figure 12. Initially, the systems engineers create a preliminary system model. Then, with inputs from domain engineers and other stakeholders, experienced modelers define the Viewpoints that express the aspects of interest to the stakeholders. For example, a Power Equipment List (PEL) Viewpoint can be defined that exposes the power characteristics to power subsystem engineers. Systems engineers then create View definitions that conform to the defined Viewpoints as the starting point of collaboration with domain engineers. Continuing the PEL example, systems engineers

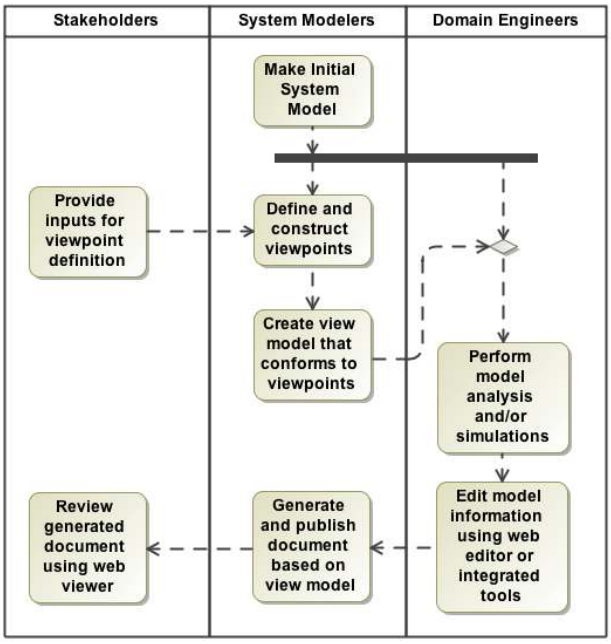


**Figure 12**. Simplified Workflow

may specify a View that only imports the avionics model elements, resulting in a PEL for the avionics subsystem. Domain engineers then take this information and do a more detailed analysis of the power characteristics (perhaps adding time based loading and discharging) that requires updates to the system model. The updates can be pushed back into the MBEE federated repository via web editors or directly through an integrated tool. The systems engineers then create a document View model (e.g., a requirements or architectural description) that is used as the vehicle of communication with other stakeholders such as project management. The review process then follows the typical document review processes with the only difference being that rather than making changes directly to the document, changes are made to the system model and the document regenerated. Not captured in Figure 12 is the iterative nature of collaboration and document generation, as model changes from one domain may necessarily impact other domains, which requires additional collaboration cycles.

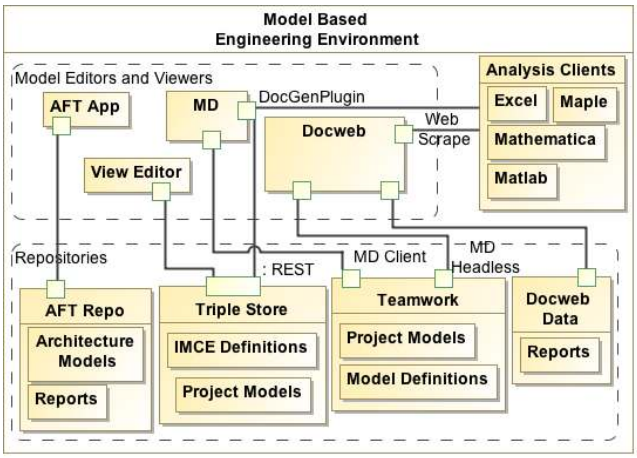


**Figure 13**. Current MBEE Components

## 5. Realizing Software and Applications

At JPL we have developed several tools and applications which implement the first version of this enterprise environment. An overview is shown in Figure 13. While these tools were constructed in an exploratory fashion, many projects have already incorporated them in their document generation workflows as they adopt MBSE practices. In particular, the Ops Revitalization project, sponsored by MGSS, generates all of its architectural documents from models using this framework and software that supports it.

*DocGen*

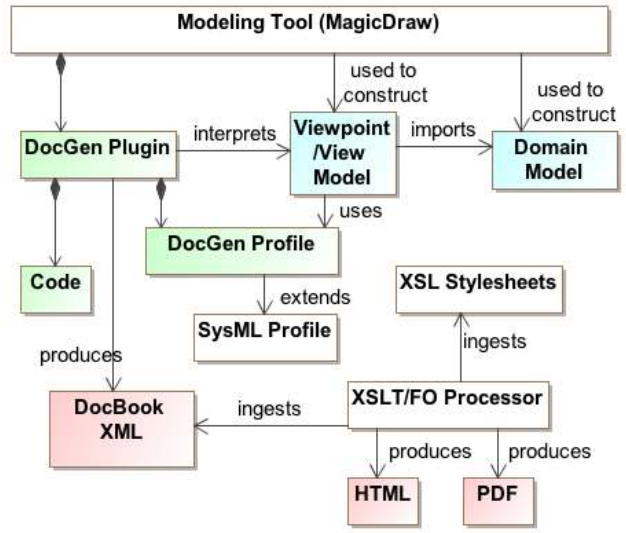


**Figure 14**. DocGen Components

**Table 12.1. Mission Service Architecture Framework Key Definitions**

| Name | Definition |
|---|---|
| Agreement | An **agreement** is a contract between 2 or more operational entities (systems, organizations, or services) that constrains how the parties will interact. |
| Activity Timeline | An **activity timeline** is a *continuous timeline* of activities, the *value* over a time interval being its assigned name (an *activity* is a named interval in time). |
| Actual Command Timeline | An **actual command timeline** is an *actual timeline* that is also a *command timeline*. It is a *command timeline* actually sent to the system being controlled by the control system. |
| Actual Measurement Timeline | An **actual measurement timeline** is an *actual timeline* that is also a *measurement timeline*. It is a *measurement timeline* actually received by the control system from the system being controlled. |
| Actual Timeline | An **actual timeline** represents data at control system interfaces of a system being controlled. A control system has access to only two actual timeline categories regarding the system being controlled: *Measurement timeline* and *command timeline*. The state of the real system being controlled is generally unobservable. Nonetheless, a simulation of the system being controlled can produce *actual state timelines* that are observable by testers. |
| Capability | A **capability** is an ability to perform a function or set of tasks based on expertise and capacity. |
| Characterizes | **Characterizes** is the relationship between a System Model and a System that asserts the System Model as a representation of the related System. |
| | A **command timeline** is a *discrete timeline* that represents the information received (or may be |

**Figure 15**. Example of a generated table of key term definitions for Ops Revitalization's Mission Service Architecture Framework

DocGen is a plugin for the MagicDraw [13] modeling tool used at JPL. The major components involved and the artifacts produced are shown in Figure 14. It provides a profile that implements the Viewpoint method specifications described in Section 3 and the capability to parse and execute Viewpoint models constructed using this profile. As shown in Figure 1, it uses existing UML import semantics to indicate which model elements should be imported by the View, which would then be passed to the Viewpoint it conforms to.

An activity model captures the Viewpoints method, where a sequence of stereotyped actions specify how to analyze, transform, and present the elements imported from the View. All stereotypes are defined in the DocGen profile as part of the DocGen plugin, and the activity is essentially the behavior of the Viewpoint. An example is shown in Figure 16. This simple Viewpoint results in a View that is an ordered list, where each list item would show the documentation of some model element that is an "Essential" class, which can be found under the namespace of elements imported from the View that conforms to this Viewpoint. The stereotypes on these actions effectively map to the rules, analyses, or transformations for that Viewpoint. For example, filtering actions can be interpreted as a rule that only certain elements that pass a test will be shown in the View. Since we are using UML activities to model the method, we have reused certain UML elements like Fork, Join, and Merge to represent actions with those same semantics. Given a library of these actions, one can then build up a library of Viewpoints for specific documents. These Viewpoints would essentially become the document templates. When one wants to generate a document from a model using a specific template, one can simply create a conforming view that imports the desired model elements as arguments to the template.

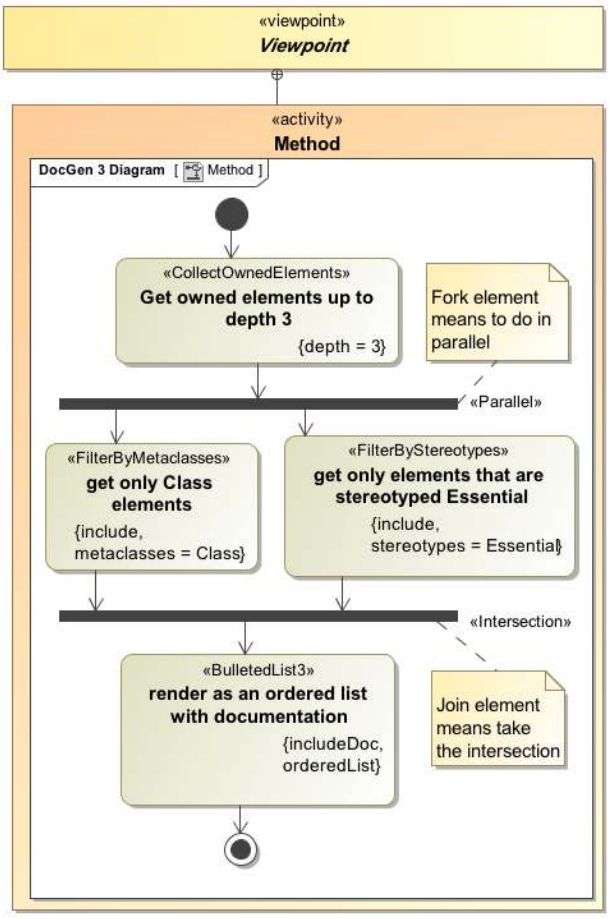


**Figure 16**. Viewpoint Method that generates a list of model elements and their documentation

The library of actions can include any type of analysis or transformation relevant to the organization. We have found that the most basic actions include following model relationships or properties to other elements, filtering collections of model elements by metaclasses or stereotypes, running custom analyses and validation rules, and displaying tables,

paragraphs, lists, or images. One very common viewpoint is generating a table of model elements and their documentation, whose resulting HTML output is shown in Figure 15. More sophisticated transformations can include parsing a model structure like composition or inheritance trees into graphs for further processing. The stereotypes are defined with tags that provide options that are relevant to that action, for example, depth, include or exclude flags, etc. Example of these tags are also shown in Figure 16. Projects can also add actions that can call user specified scripts that contain more project specific rules for checking the model and constructing a custom display, like doing mass or power rollups for flight systems and reporting on errors found.

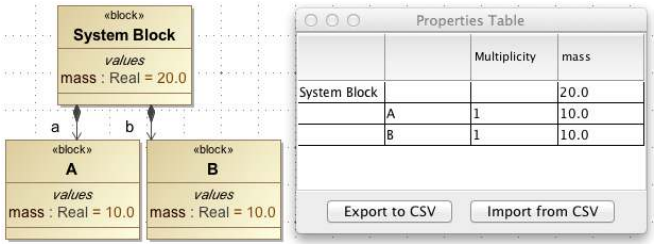


**Figure 17**. An Editable Table

Since a document is composed of Views, Figure 3 shows how a View hierarchy can be modeled and interpreted. From the "Root View" package, which denotes the root of a document, linked list semantics are used to indicate the first child View and subsequent Views, where by default each View will be interpreted as a section in the resulting document. Given a library of Viewpoints, one can easily string together the View model and conform each View to an appropriate Viewpoint according to the needs of each document.

Currently, the most common use case for DocGen is to output the results of viewpoint execution into DocBook XML, but given the right specifications it can also show editable tables within the modeling application and publish editable views to the web. In the case of tables, since the content in table cells ultimately come from some property of model elements, DocGen provides an edit mode - instead of rendering a static table, a pop up table is displayed where users can directly edit those model properties. Figure 17 shows an example of editing mass properties of a system composition.

As this illustrates, Views are not restricted to being parts in a static document. They can be outputted in any format, limited only by the format and presentation options specified in the Viewpoint. A dynamic View like the editable table significantly eases collaboration with domain engineers and other stakeholders who provide inputs to the model.

It is important to note that the DocGen implementation is not the only way to realize the View-Viewpoint paradigm. Although we primarily work with SysML models, the Viewpoint specification can theoretically be implemented in any language and a set of rules and transformations defined for the target language. The steps in the Viewpoint method can operate on a heterogeneous set of models, such as ontological models, CAD models, as well as SysML models, as long as there exists a unified way of describing these models.

*View Editor*

Complementing DocGen are various web applications that facilitate communication with domain engineers and stakeholders. The View Editor is an example where domain engineers can update the model online through HTML formatted Views that are specific to their discipline. Figure 18 shows how this capability is built around the existing DocGen plugin by outputting the interpreted View information in different formats. Instead of outputting to DocBook XML, DocGen can instead serialize and package the same information to a database through a REST interface. By having the software keep track of where the content of a View comes from in the model repository, users can update specific parts of the model without having to know the details of how the model is put together or even open the modeling application. Figure 19 shows the web page of a View, where users can directly edit the contents. The View tree is also shown on the left as a navigation pane for each of the document's sections.

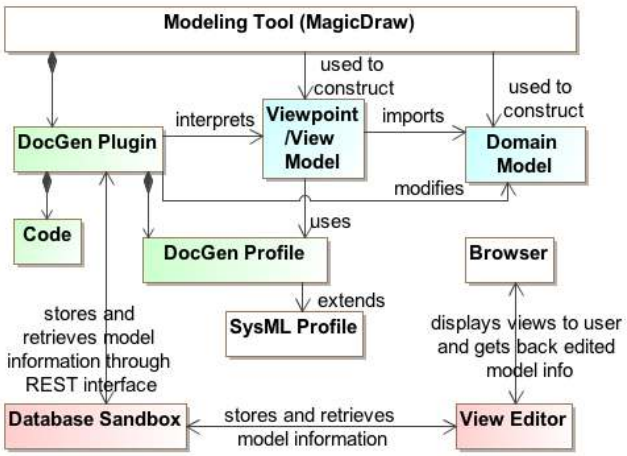


**Figure 18**. View Editor Components

To achieve this, DocGen packages and upload subsets of the model and View information to a database that the View Editor operates on. Users can then update selected model information through the web that gets persisted in the database. Cognizant system modelers will then import these changes back into the model. Currently this extra layer is necessary because of the lack of a central and accessible model repository. Imagine then, if any tool can access model information directly through a repository that houses all model, Viewpoint and View information. Without the middleman, tools can interpret Viewpoints directly and produce appropriate Views according to the formats and presentation defined for that tool. This technique can be used to integrate with existing or new analysis tools. By adjusting the View format, we are essentially defining an interface that can transmit subsets of model data back and forth with applications like Excel, Mathematica, Matlab, and more.

*DocWeb*

To facilitate document generation and review, we have developed a web application for requesting, scheduling, and archiving artifacts generated from the model. Again, the web interface and necessary additions are built around the core DocGen plugin and the Viewpoint/View framework, as shown in Figure 20. The output format is DocBook XML, which can then be transformed into HTML and PDF. CSV files from any relevant tables can also accompany the generation, and possibly more in the future. These artifacts are archived and tagged with a timestamp and can be retrieved through a web interface, as shown in Figure 21. Options for on demand generation or scheduled generation, like nightly or

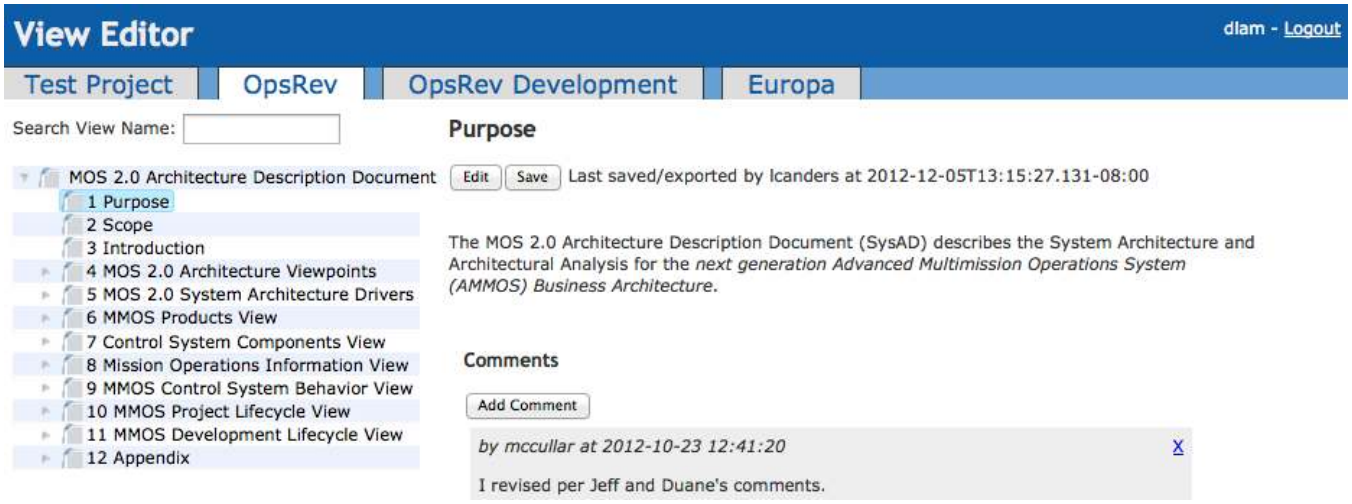


**Figure 19**. View Editor Example - A view showing editable text and view navigation on the left

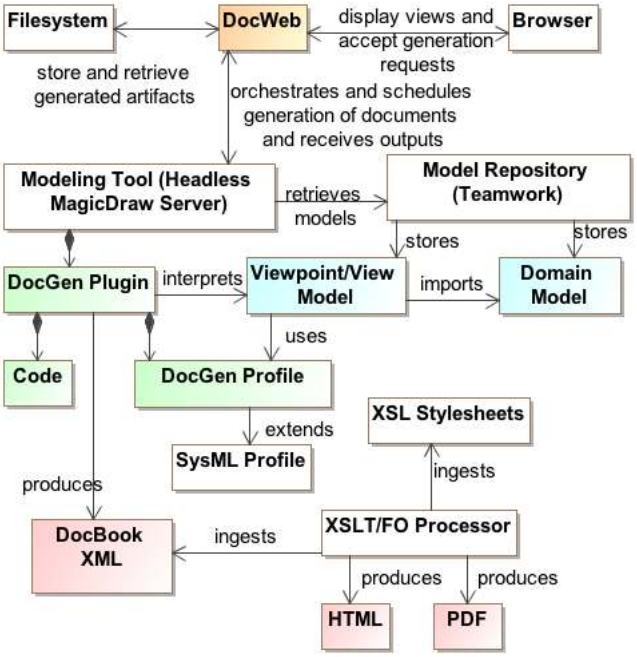


**Figure 20**. DocWeb Components

weekly, allow system engineers to monitor the general state of the model and documents and be alerted in a timely manner if any problems arise, such as failed generations or failed validations. Since the model repository houses both the system models and Viewpoint models that describe how to create Views, the entire generation chain can be automated to ensure that documents will always be up to date and consistent with respect to the model, no matter how frequently the model gets updated.

## 6. Conclusion

As MBSE becomes mainstream, the need for a more automated and streamlined approach to model based document generation increases. We have extended the SysML concepts of View and Viewpoint in order to create a foundation to

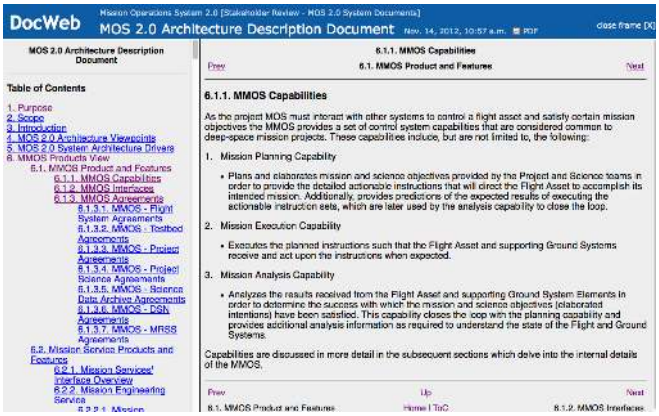


**Figure 21**. DocWeb Example - A generated document with navigation on the left and section content on the right

address this need. This allows systems engineers to use Viewpoint models to describe how to extract, analyze, and present specific information from the system model to stakeholders and domain engineers. In addition to generating just static artifacts, the format option in the extension also supports a way to specify integration with other software that can manipulate model information. We envision that a model based engineering environment with a central repository of model and Viewpoint information will be the key to integrating all the pieces needed to execute successfully in a model based project. We have developed software like DocGen, View Editor, and DocWeb to pave the way to realizing this vision.

## Acknowledgments

The work described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

## References

[1] Object Management Group, "OMG Systems Modeling Language (OMG SysML$^{TM}$), version 1.3," OMG,

Tech. Rep. OMG document number formal/12-06-02, June 2012.

[2] M. Jackson, C. L. Delp, D. Bindschadler, M. Sarrel, R. Wollaeger, and D. Lam, "Dynamic Gate Product and Artifact Generation from System Models," in *Proceedings of Aerospace Conference*. Big Sky, Montana: IEEE, 2011.

[3] D. Bindschadler, C. L. Delp, and M. McCullar, "Principles to Products: Toward Realizing MOS 2.0," in *Proceedings of SpaceOps Conference*. Stockholm, Sweden: AIAA, 2012.

[4] T. Bayer, S. Chung, B. Cole, B. Cooke, F. Dekens, C. Delp, I. Gontijo, and D. Wagner, "Update on the Model Based Systems Engineering on the Europa Mission Concept Study," in *Proceedings of Aerospace Conference*. Big Sky, Montana: IEEE, 2013.

[5] T. Bayer, M. Bennett, C. L. Delp, D. Dvorak, J. S. Jenkins, and S. Mandutianu, "Update - concept of operations for integrated model-centric engineering at JPL," in *Proceedings of Aerospace Conference*. Big Sky, Montana: IEEE, 2011.

[6] DocBook Technical Committee, "DocBook 5," OASIS, Tech. Rep., 2009, http://docbook.org.

[7] MGSS, "Advanced Multi-Mission Operations System," http://ammos.jpl.nasa.gov/.

[8] OPFM, "Europa Jupiter System Mission," https://opfm.jpl.nasa.gov/europajupitersystemmission-ejsm/.

[9] OMG Systems Engineering DSIG, June 2012, http://www.omg.org/news/meetings/tc/ma-12/info.htm.

[10] ISO/IEC/IEEE, "Systems and software engineering - architecture description," ISO/IEC/IEEE, Tech. Rep. ISO/IEC/IEEE 42010, December 2011.

[11] Europa Study Team, "Europa Study 2012 Report," National Aeronautics and Space Administration, May 1 2012.

[12] S. Chung, T. Bayer, B. Cole, B. Cooke, F. Dekens, C. Delp, and D. Lam, "Model-Based Systems Engineering Approach to Managing Mass Margin," in *5th International Workshop on Systems and Concurrent Engineering for Space Applications*. Lisbon, Portugal: ESA, 2012.

[13] NoMagic, "MagicDraw," http://www.nomagic.com/.

## BIOGRAPHY

**Christopher Delp** *is the Systems Architect for Ops Revitalization task in MGSS and a Lead Systems Engineer for MBSE on the Europa Mission. He is a member of of the Systems Behavior and Architectures Group at the Jet Propulsion Laboratory. His interests includes Systems and Software Architecture, applications of Model-Based Systems Engineering , Model-Based Analysis and Enterprise Engineering Systems. He earned his M.S. and B.S. degrees from the U of A in Systems Engineering.*

**Doris Lam** *is currently a Software Systems Architect working in the Model Based Engineering Environment team at JPL. She earned her B.S. in computer science from UCLA in 2008 and joined JPL after graduating. She has worked on various UML and SysML modeling projects and software modernization tasks for the ground system.*

**Elyse Fosse** *is a Software Systems Engineer for the Ops Revitalization task in MGSS. She also develops ground system cost models for deep space and Earth missions. She is also a member of the Multimission Ground Data System Engineering group at the Jet Propulsion Laboratory. Her interests include software and systems architecture, applications of model-based system engineering, and cost model implementation and analysis. Elyse is also a part of the INCOSE Space Systems Working Group's entry into the Model Based Systems Engineering Grand Challenge. Elyse earned her M.A. in Applied Mathematics from Claremont Graduate University and her B.S. in Mathematics from the University of Massachusetts Amherst.*

**Cin-Young Lee** *is a Senior Software Engineer in the Mission Information Systems and Technology Development Group at the Jet Propulsion Laboratory. He earned his Ph.D. from Caltech.*