The JPL Systems Environment

Robert Karban*[a], Marie Piette[a], Eric Brower[a], Ivan Gomes[a], Emilee Bovre[a], Myra Lattimore[a], Blake Regalia[a], John Carr[a], Chad Harris[a], Christopher Delp[a], Cin-Young Lee[a]

[a]Jet Propulsion Laboratory California Institute of Technology Pasadena, CA 91109, USA

*robert.karban@jpl.nasa.gov

Abstract

The Jet Propulsion Laboratory (JPL) carries out a wide range of robotic space missions that require an immense interdisciplinary engineering effort.

JPL has initiated a transformation in the way it develops and delivers engineering tools to these missions, known as “Flight Projects”, and the Flight Project Engineers. This transformation sets the goal of unifying previously disjointed engineering tools and services into an environment capable of allowing broad collaboration, accurate authoritative engineering models, sophisticated analysis, and simulation. The JPL Systems Environment provides this unified function for the threads of information that enable the system’s representation of a digital twin of the Mission.

The JPL Systems Environment facilitates a formalized application of systems engineering by leveraging models for requirements, design, analysis, verification and validation activities from conceptual design, to development, and through operations. Flight Project Engineers can develop their system models for qualification and certification across a portfolio of engineering tools. The environment is a Multi-Paradigm Modeling environment that has support and compatibility for languages such as OWL, SysML, BPMN, Modelica, and Python.

The JPL Systems Environment is designed to provide information in a connectable way for other platforms, services, and tools. This is done without the overhead of traditional manual exchanges of information. Instead, the JPL Systems Environment reduces redundancy and increases the value of engineering products. Engineers can collaboratively construct and analyze the precision products needed to develop missions and systems at JPL.

Following a development pipeline paradigm, the environment orchestrates the workflow of collaboration and data de-confliction, requirements and design, analysis, and publication of engineering products. Several Flight Projects (e.g. Europa Clipper, Europa Lander, Mars 2020) have adopted the environment and tailored it to their needs.

Keywords: MBSE, MPM, MPM4CPS, Modeling Environment, MBEE, Model Based Engineering

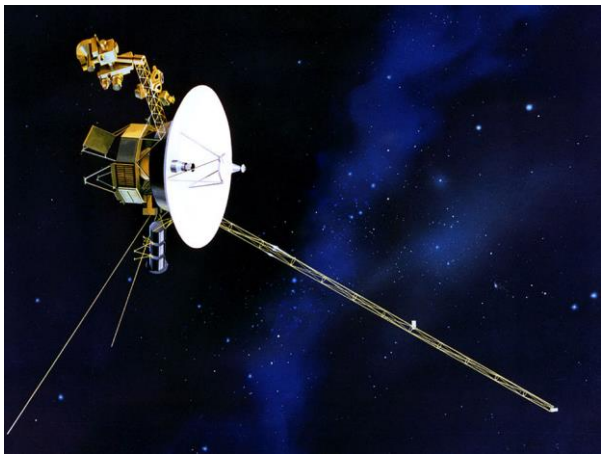
1 Context

The **Jet Propulsion Laboratory** (**JPL**) is a federally funded research and development center managed for NASA by Caltech.

Today, JPL continues its world-leading innovation while implementing programs in planetary exploration, Earth science, space-based astronomy, technology development, and the application of its capabilities to technical and scientific problems of national significance. JPL technology originally developed to enable new missions is also applied on Earth to benefit our everyday lives.

Among the laboratory's major active projects are the Mars Science Laboratory mission (which includes the *Curiosity* rover), the Mars Reconnaissance Orbiter, the *Juno* spacecraft orbiting Jupiter, the NuSTAR X-ray telescope, the SMAP satellite for earth surface soil moisture monitoring, the Spitzer Space Telescope, and Voyager 1 & 2. It is also responsible for managing the JPL Small-Body Database, which provides physical data and lists of publications for all known small Solar System bodies.

JPL has over 6,000 employees with approximately 4,000 engineers who perform various engineering and systems engineering tasks with growing complexity using a multitude of tools directed by a rich set of engineering rules and processes.



Voyager 1&2 (1977)

Mars Science Laboratory (2012)

As JPL develops and deploys more autonomous and complex robotic systems, it requires transformations across the lab that enable easier collaboration and information sharing. An engineering modeling environment strengthens end-to-end mission capabilities and accelerates the infusion of new technologies into future missions.

2 Background

2.1 Systems Engineering Process

A typical system engineering process (Figure 1) that follows the international standard ISO-15288 [3] to design and develop a system can be divided into four key processes: 1) requirements development, 2) architectural design, 3) technical evaluation, and 4) synthesis. These four processes are conducted iteratively over several lifecycle phases in concurrence with one another and culminate in regular reviews.

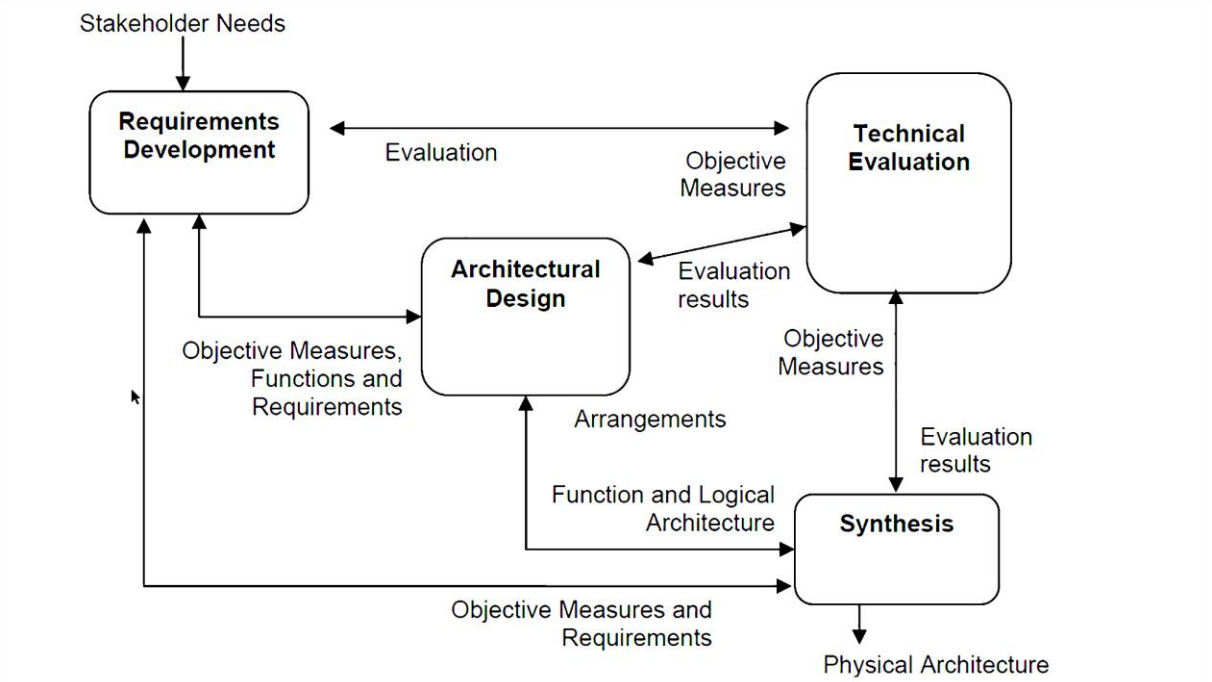


Figure 1. The ISO-15288 Systems Engineering Process [3]

The JPL Systems Environment is designed to be process agnostic but the atomic methods of the ISO-15288 systems engineering process is a subset of all functionality offered by the JPL Systems Environment.

2.2 Model-Based Engineering

Model-Based Engineering (MBE) is the formalized application of modeling to support system requirements, design, analysis, verification, validation, and documentation activities. MBE spans the entire lifecycle of complex systems engineering projects.

INCOSE [4] adopts a V-model for its advocated systems engineering process that is to be used for defining and developing arbitrary systems. The traditional systems engineering's V-Process needs to be contextualized to the model-based paradigm. The JPL V-Process (Figure 2) introduces different models

for each step in the process. The goal is to use models to enable traceability of the artifacts along the legs of the V, implying to have traceability between the models themselves on the vertical axis, but also between the left and right leg of the V on the horizontal axis.

Project Models are concerned with the project as a whole across all Systems Engineering disciplines such as Flight System Systems Engineering or Mission Operations Systems Engineering. Project Models should be traceable to the Science Models and requirements they respond to. Science models typically explore science cases through the analysis of simulated mission data. The purpose is to provide a quantitative assessment of the extent to which a mission will be capable of addressing key scientific questions. The Mission V&V validates if the mission can deliver the expected scientific data. Specific System Models, which have to be mutually consistent, address how the system responds to the project requirements and model. Subsystem Models address domain specific questions, for example mechanical and electrical engineering, such as CAD models or electrical function lists.
Subsystem Models can be refined by for example VHDL (Very High-Speed Integrated Circuit Hardware Description Language) which is a hardware description language for electronic design automation, or Simulink for control loop design. Those are models very close to implementation and should be traceable to subsystem models.
The JPL Systems Environment currently supports project and system models, their associated modeling artifacts that are created early in the development process, as well as related high-level analysis. Specifically, these artifacts are represented in Level 2 Project Requirements, Level 3 System Requirements, Level 2 Project Verification & Validation (V&V), and Level 3 System V&V.

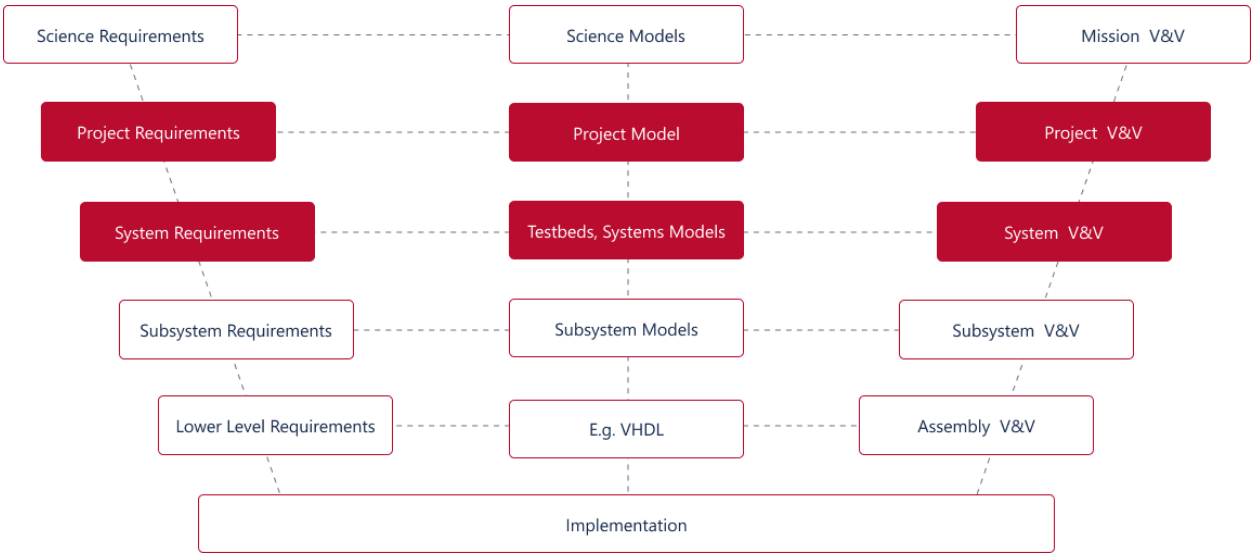


Figure 2. The V-model of the JPL Systems Engineering Process

2.3 Digital Engineering Infrastructure

As previously mentioned, engineers live in a diverse landscape of engineering applications and services for a variety of information models in different domains, including Systems, Software, Electronics, and Mechanics, which are often implicitly connected [10]. In order to successfully make use of a MBE

paradigm, these models and their information must be explicitly formalized, connected, and managed. The systems engineering data, such as architectural models, need to be integrated with the remaining engineering artifacts. The resulting conceptualized information structure is a graph, as shown in Figure 3. The JPL Systems Environment enables to manage this graph of information with their sources and targets as well as the connections, and their versions. Data is synchronized between different sources still maintaining the single source of authority where a piece of information has a clearly designated source but can be replicated in a version-controlled way. Version control on the source data, target data, and the connection allows for enhanced traceability and configuration management as design and analysis iterate throughout the project lifecycle; engineers can know when the source data is changed and decide when to accept that change for further analysis. The JPL Systems Environment architecture provides a number of integrated web-services which allow clients to access, author, process, and publish data and keep everything configuration managed. For example, system architecture elements are traced to requirements but are also used to perform analysis (e.g. simulation) and capture how a software implementation responds to the architecture. System architecture elements can also propagate into Product-Lifecycle-Management (PLM) systems where they map to Mechanical (MCAD) and Electrical (ECAD) CAD models. The project management schedule refers to specific information in the system architecture and PLM data to keep them consistent and checkable.

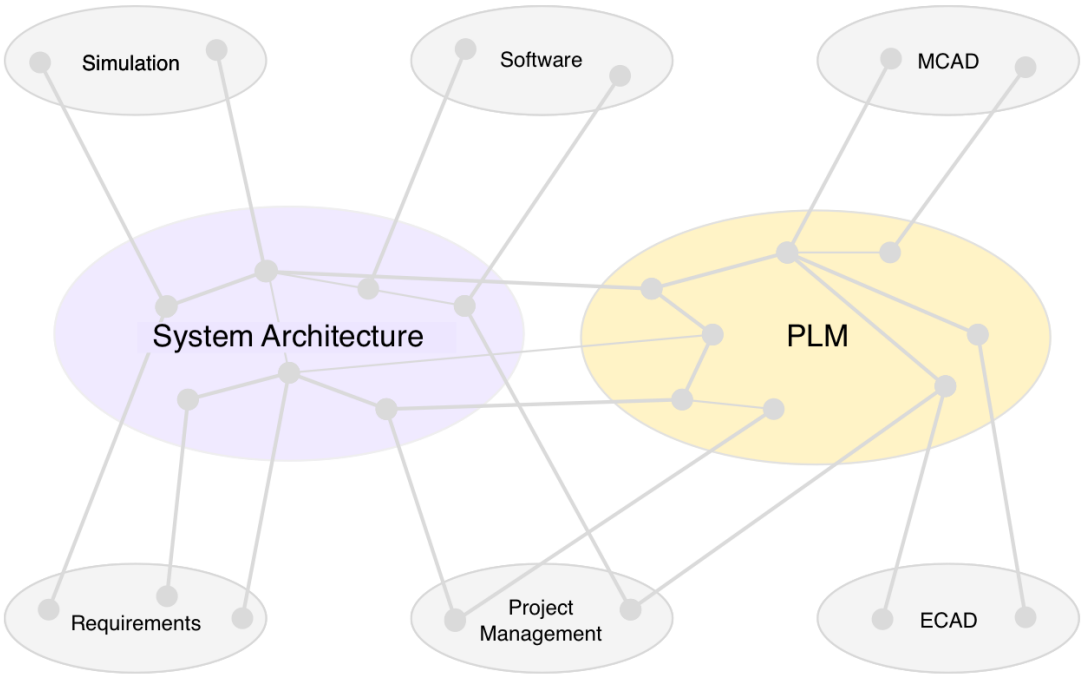


Figure 3. Graph Structure of Information

2.4

3 Introduction

3.1 Current State of System Engineering Practice

Engineering of complex systems relies heavily on the use of models addressing concerns from different domains such as mechanics, optics, software, and control. Models of a wide range of formalisms, e.g. CAD (Computer Aided Design), FEM (Finite Element Model), and custom software models, are ubiquitous in domain-specific engineering. Each of these models are generally specialized for the problem they are addressing and the level of abstraction or specification reflects that focus. An example could be a systems engineering model may treat all subsystems as “black boxes” while focusing on specifying the interfaces by which the subsystems interact, while a subsystem engineer may only focus on the behavior of their component. Some of the models may conform to the same metamodel, e.g. SysML, BPMN, and UML, some have standardized mappings between metamodels, e.g. Modelica, Simulink, and SysML via SysML Extension for Physical Interaction and Signal Flow Simulation [56], and still others are largely unique, e.g. custom software models or domain specific languages. Document-based artifacts often must describe multiple models or explain how their content is related. The latter is typical in systems engineering. These documents must reflect the three dimensions of Multi-Paradigm Modeling for Cyber Physical Systems: (1) multiple levels of abstraction, (2) multiformalism modeling, and (3) meta-modeling [54]. This methodology relies on formal models rather than documents to act as the source of truth for systems engineering information. These system-level models can be used to tie together and federate different subsystem (e.g. mechanical, thermal, electrical) models, and can also be used to generate different documents that are needed by the stakeholders.

In systems engineering practices utilized by JPL and other organizations, system design and analysis have been historically performed using a document-centric approach [40]. In a document-centric approach, stakeholders produce a number of documents that represent their respective views on the mission or system under development. Given the ad-hoc, disparate, and informal nature of natural language documents, these views can quickly become inconsistent. Currently, dependencies between these views [2] are often implicit or informally defined in supplementary documents (Figure 4). Additionally, documents are often accompanied by a variety of engineering models that are siloed. It is often difficult to trace provenance across the different, distributed sources of information and verify their consistency. A significant amount of time is consumed by engineers searching for up-to-date information from other stakeholders.

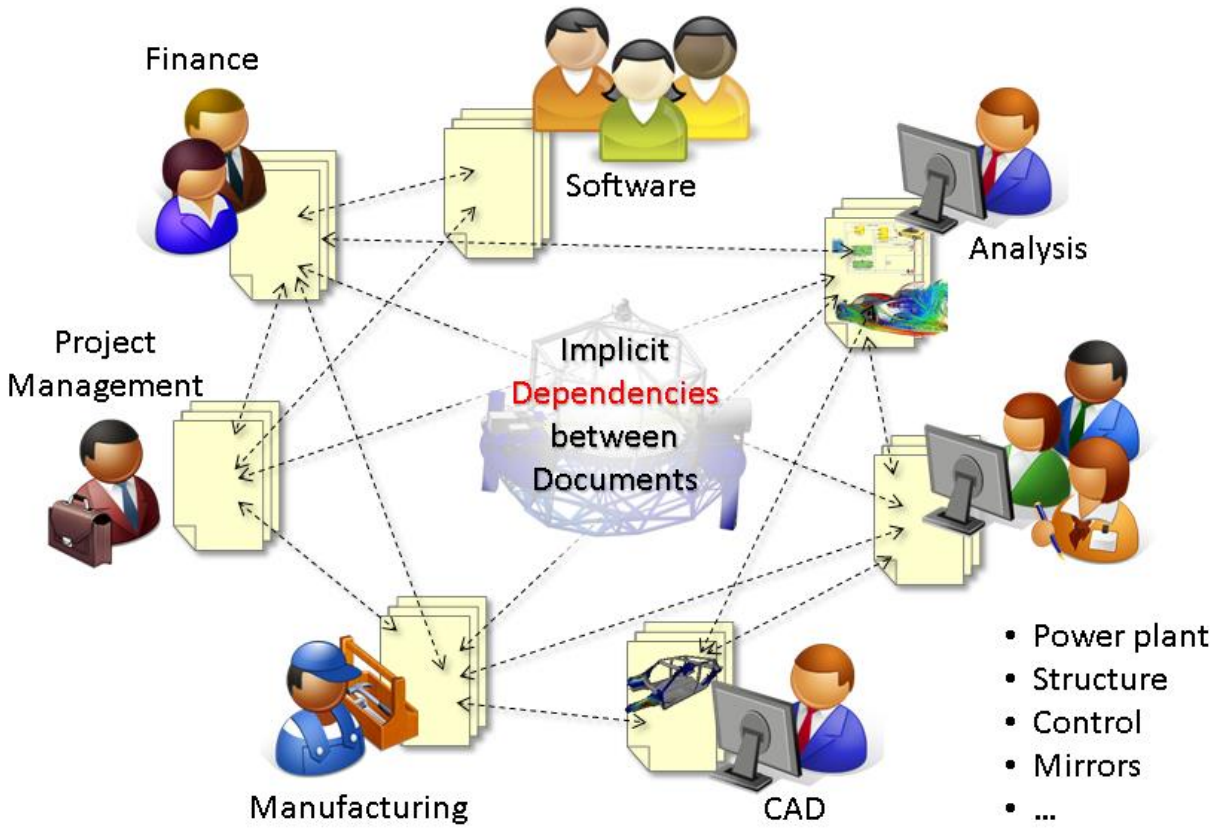


Figure 4. Implicit dependencies between system engineering documents [53]

In a document-centric approach, rigor in engineering work is lost in the transition from engineering design and analysis to engineering documents. Once the documents are delivered, the engineering portion of the work is disconnected from the resulting artifacts. For example, in the following thought experiment, Simulink model data and system requirements are merged at the point of document writing and never represented in a managed engineering environment (Figure 5). The design represented in the Simulink Model responds to requirements captured in DOORS and is usually manually managed with some kind of traceability matrix in a document without managing the configuration of requirements or design elements. Furthermore, elements of the design and requirements are referenced in the engineering documentation but they are neither configuration managed nor formally traceable to the engineering source.

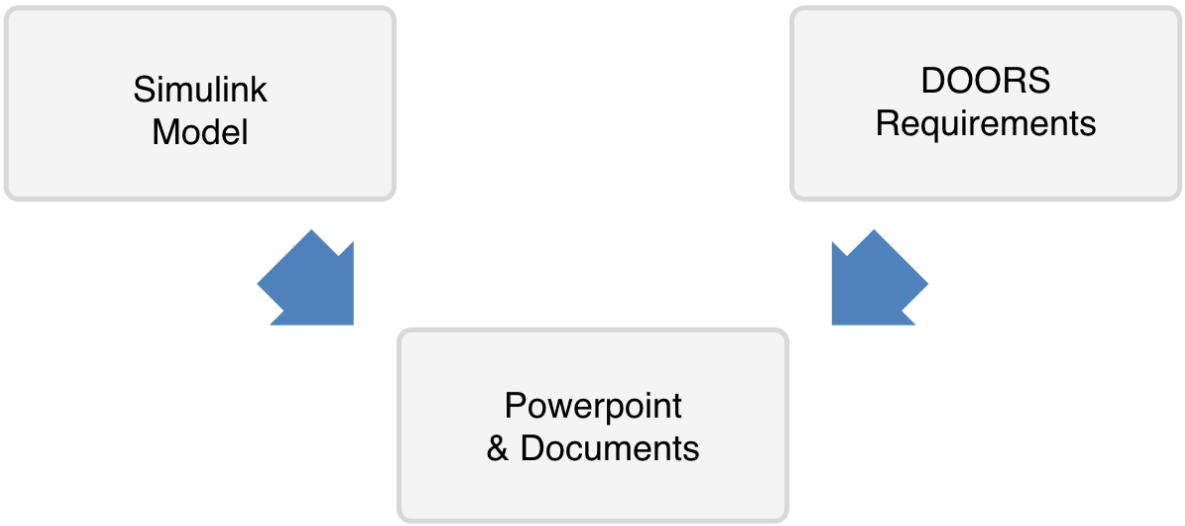


Figure 5. Models and Requirements are used to produce Presentations and Documents

Numbers, statements, and conclusions are now siloed and disconnected from the original engineering data, resulting in the last author serving as the source of truth, and implying a loss of value of the intellectual property. It requires an immense overhead to maintain consistency and perform impact analysis as one must arduously rediscover the previous engineering work instead of immediately progressing to the next engineering task. Using documents to inform engineering design or analysis is not rigorous because it is difficult to audit the original source of data and decisions that were made.

Historically, missions fail more often due to decisional failures than component failures [42]. The document-induced disconnect contributes significantly to those failures because often the required information to make a decision is incomplete, obsolete, or not traceable. There is a paradigm mismatch between the engineering work, which is often supported by models, and the deliverables in the form of documents. Moreover, most stakeholders and consumers are not modelers; however, there is the desire to know with confidence that the captured models and the documentation are mutually consistent. Engineering of complex systems necessitates high levels of collaboration, and in analogy to Google Docs, the questions that arise with this collaborative paradigm also occur for technical projects.

The JPL Systems Environment strives to make those implicit dependencies explicit and configuration managed. The environment promotes a best of breed integration at environment level and not point to point; this allows all applications and services to integrate on our terms and not on vendor terms as it is often enforced by monolithic platforms. The requirement is integration and not replacement to enable digital threads in a web-service centered architecture.
Having the Open Model Based Engineering Environment (OpenMBEE) [5] software infrastructure at the core, allows to digitally connect the engineering documentation with the engineering models by embedding engineering data as linked data directly in the narrative in a version-controlled way. In the case of the Simulink model and the requirements housed in DOORS [51], not only is each source data versioned in a configurable way, but they can be formally linked to each other and linked to the

document. Versioning the links themselves also allows for extra traceability. Using version control across all sources ensures that the documentation that is produced for narrative and reporting, will be traceable to the identified versions of the source data. Configuration management can also help ensure when and how things are updated and synced across resources. This key novelty allows us to keep documentation and engineering works mutually consistent, up to date, and configuration managed.

3.2 Research Questions

Based on the problem described in the current state of systems engineering, the following research questions are posed:

1. How can models and documentation be kept mutually consistent and correspondent?
2. How can modeling languages be used collaboratively?
3. How can the live collaboration and connectivity paradigm work in the context of rigorous engineering information?
4. How can engineering work be preserved and remain traceable to documentation for impact analysis to assess the consequences of changes in the model on documentation?

3.3 Hypotheses

In pursuit of answering the given research questions, we formulate the following hypotheses to be validated or falsified experimentally. We propose an environment *E* where:

H1 As opposed to systems with a discontinuous document publish and review process, *E* reduces the user effort to produce consistent engineering documents.

H2 In a multi-user, multi-language engineering context, *E* bridges the gaps between documents and model elements across tool boundaries.

H3 *E* is flexible in terms of the ‘model-hardness’ it tolerates during the ongoing maturity of a project.

H4 All versions of any deliverable that is derived from *E* is stored along with its comprehensive traceability metadata, which also enables on-demand reproducibility.

4 Method

JPL missions are developed using a wide variety of software applications and services. The JPL Systems Environment aims to provide an end-to-end connected environment for these tools to work together in order to support JPL's various projects. Services and tools must be able to support a wide range of analysis and integration capabilities. Integrated lifecycle support for these tools is provided, which includes configuration management, archival, business process implementation, review support, and interconnectivity and propagation of data between new and existing tools.

The JPL Systems Environment is an MBE ecosystem composed of applications and services relevant to systems development and analysis. The environment creates and maintains relationships between heterogeneous data sources.

The following methods address the research questions which are elaborated in detail on the following pages:

1. Federated and multi-language transclusion
2. Incorporation of unstructured data into structured data
3. Collaborative workspace for model-based documents
4. Systems development pipeline
5. Digital threads
6. Engineering cookbooks

The first three methods are implemented by the Open Model Based Engineering Environment (OpenMBEE) [5] software infrastructure including the Model Management System (MMS), the View Editor (VE), the Model Development Kits (MDKs), and the Platform for Model Analysis (PMA). MDKs are the integration mechanism of OpenMBEE for engineering applications to allow them to create, update and delete data in a model consistent manner. The systems development pipeline of the JPL Systems Environment enables a continuous, iterative, and incremental engineering lifecycle process, analogous to a modern software development pipeline. Digital threads are supported by a flexible web and service based architecture to integrate data, services, and tools, which support a vast amount of engineering workflows. Engineering cookbooks capture best practices and lessons learned by providing goal oriented guidance on building systems models with the available tooling.

4.1 OpenMBEE

OpenMBEE [5] is the core open-source portion of the JPL Systems Environment which provides a platform for modeling applications acting as clients and a web front-end serving as a multi-tool, multi-language (e.g. OWL [6], SysML [7], BPMN [8], Modelica [9], MATLAB [41], Mathematica [11], Python [12]), and multi-repository integration platform across JPL's engineering and management disciplines. OpenMBEE provides an infrastructure for versioning, configuration management, workflow, access control, content flexibility, continuous analysis, web application development, and web-based API access.

OpenMBEE is an integrated set of software applications and services which replaces silos of information with consistent, traceable, and precise engineering models and documents. Information is mutually consistent and correspondent. Using transclusions, the platform upgrades a document-based process with a model-based engineering environment. Transclusion is the rendering of model information inline with unstructured narrative in model-based documents. Figure 6 illustrates this concept. In the first step, a piece of engineering information, shown in blue, exists in models, processes, and documents and is initially disconnected. In the second step, transclusions are used to create linked-data documents that close the gap between model and documentation, ensuring that they are mutually consistent and correspondent. In the third step, those documents are made available in a collaborative space where engineers author, review, modify, and release those documents according to the access granted to their respective roles in each linked model.

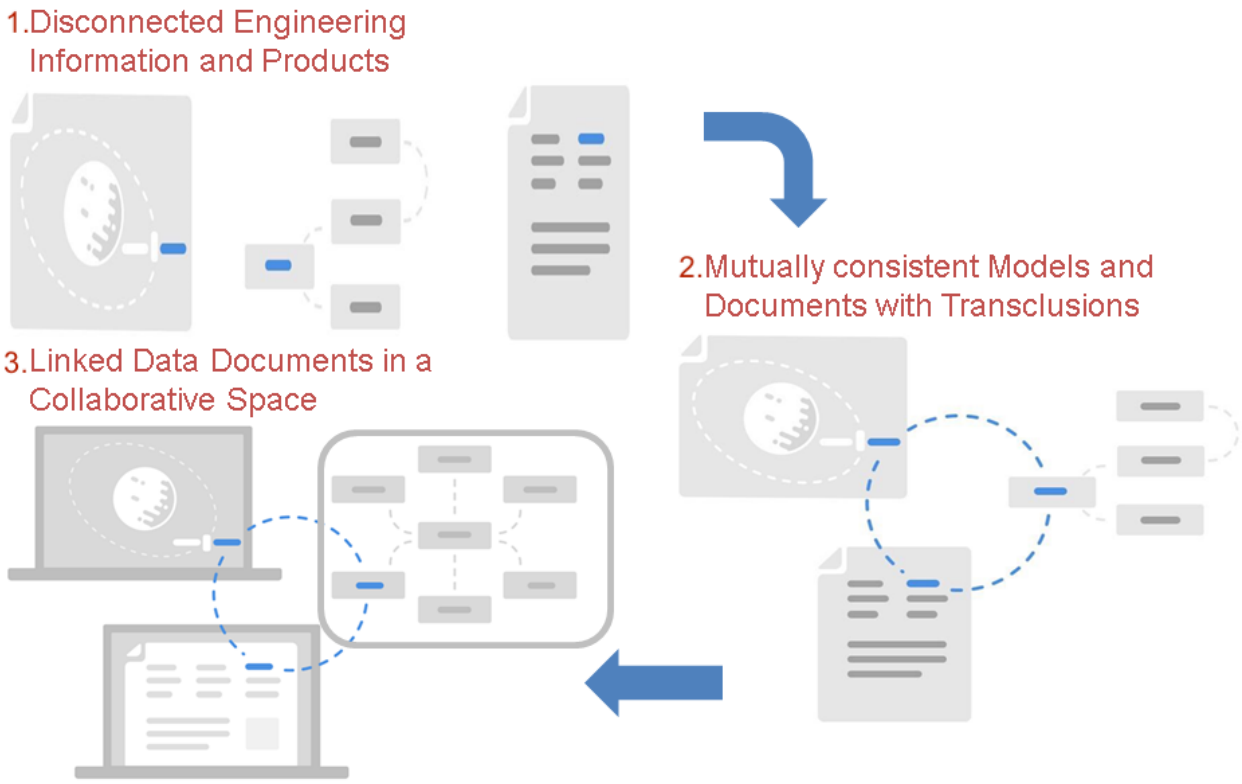
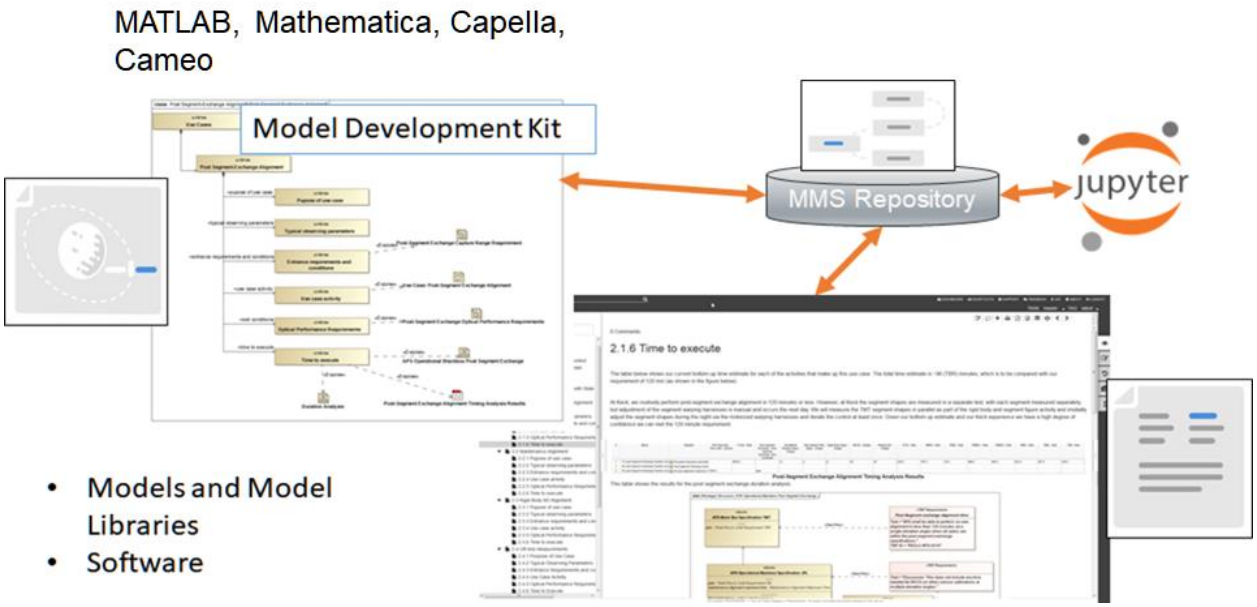


Figure 6. OpenMBEE concepts

OpenMBEE constructs linked-data documents by using the Document Generator (DocGen) query language and MDK [1] to gather relevant model data to produce views. DocGen is a module of MDK that constructs and generates views that consist of model elements collected by the DocGen query language. The model elements are sourced from different modeling languages and the views conform to the IEEE/ISO 42010 view and viewpoint paradigm [2]. The method by which data is extracted out of and/or synced with applications and services are adapted for its conventions and native workflows. For example, in the case of the Cameo Systems Modeler client changes are tracked and transparently synchronized bidirectionally with MMS while the MATLAB MDK "toolbox" allows users to integrate the provided MMS element pull and push functions directly into their MATLAB code. The data is version controlled in the MMS repository and rendered as an editable document in the View Editor web application. The model can be accessed through a rich modeling tool such as Cameo Systems Modeler, MATLAB or Mathematica, through an HTTP-based REST API [44], or an analysis platform such as Jupyter [13]. Figure 7 shows how the OpenMBEE concepts map to the implementation.



4.2

Figure 7. OpenMBEE Concept and Implementation

OpenMBEE's implementation allows the engineer to focus on technical and engineering work by concurrently editing both the model and document, thus reducing cost and time by preventing duplication of work and automatically keeping the documentation and models consistent. Sharing the documents in a collaborative web application reduces inconsistent data while improving accessibility of model data to stakeholders.

DocGen [14][15] provides a structured representation of documents while still allowing incorporation of unstructured data into structured engineering data. This mechanism dramatically lowers the barrier for modeling by enabling a process referred to as *model hardening*, by which engineers can incrementally add formalism to their models as their thinking matures from concept to design.

Transclusions can synthesize data for engineering when the information is not simply hyperlinked, but referenced in place. Therefore, the link to the authoritative source of information is preserved and maintained. The use of transclusions also provides a new means of measuring the maturity of the captured information overall referred to as *model hardness*. For example, early in the lifecycle models are often boxes and lines without semantics, have little to no scope, or are developed without external context. Those boxes and lines are gradually transformed into formal system models as abstractions are defined and the problem analysis progresses. Systems Engineering documents are largely narrative but reference model data informally, e.g. by cut and paste, which is error prone and difficult to trace. OpenMBEE addresses this problem by its transclusions, i.e. allowing the authors directly embed model data into the narrative and keep it under version control. Early lifecycle documents which are generated or related to these models typically utilize a relatively low number of transclusions. By the end of the life-cycle, the interconnection and use of models and structured data should be rich, which is evidenced by a large number of transclusions. Unstructured data indicates sections of the engineering design which require more development. A mature and structured model enables projects to define more focused rules and

policies for verification of the model. The corresponding documents therefore include more transclusions from the model in the late life-cycle. To this effect, our use of transclusions validates H2 since documents are linked to ‘living, breathing’ model data outside the bounds of any single tool in which they are created or modified.

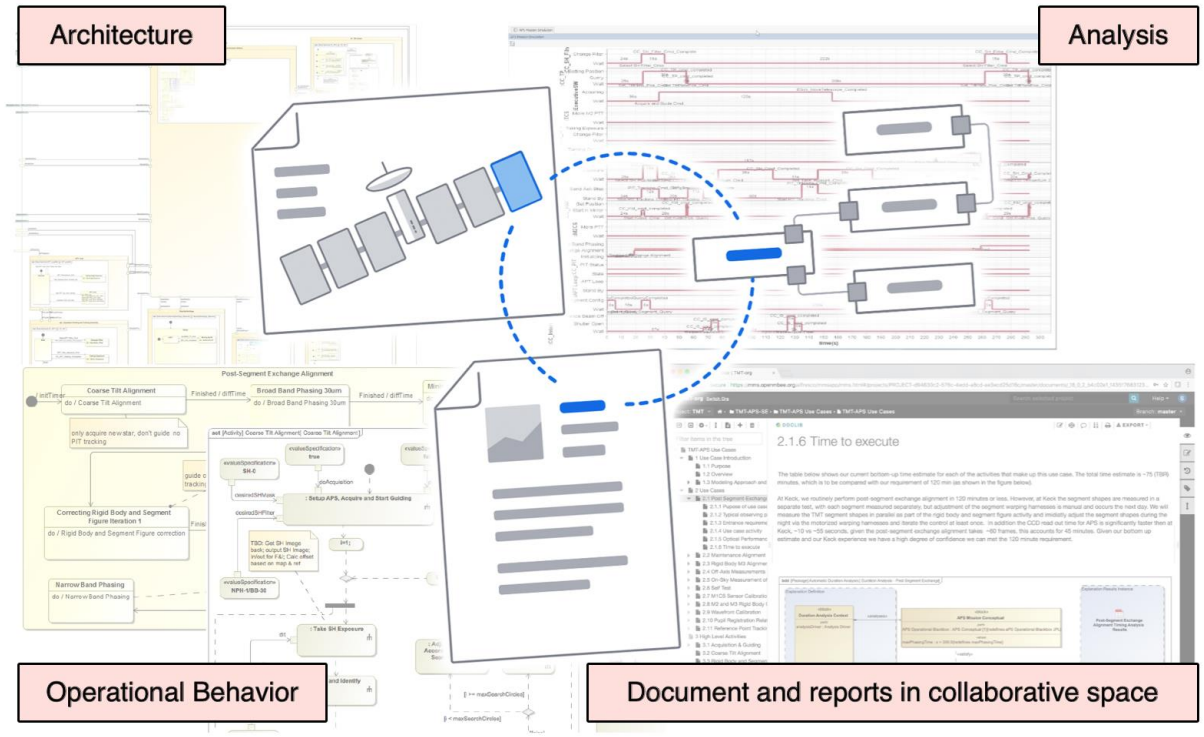


Figure 8. Thirty Meter Telescope End-To-End Digital Twin

Figure 8 shows an excerpt from the production engineering models and documents of the Thirty Meter Telescope (TMT) project [16] which uses OpenMBEE to build an end-to-end digital twin [17][18]. The TMT system model captures the architecture using SysML internal block diagrams and operational behavior with state machines and activities. The requirements are verified against the as-designed system using simulation as an analysis method (the model is fully executable) which produces timelines of states and values. The engineering documents reference information from the model and the simulation results using transclusions in OpenMBEE View Editor. Whenever the model is updated, the analysis is run and the document is republished and therefore keeping system model, analysis, and documentation mutually consistent and up to date. This automated synchronicity between the model and documentation provides the model-based analog of Continuous Integration which reduces user effort to produce consistent engineering documents, thus validating H1.

This pattern supports the view/viewpoint paradigm by querying the desired elements in the model and creating a view meant for the specified stakeholder. Figure 9 describes the model-based document generation process within the OpenMBEE platform.

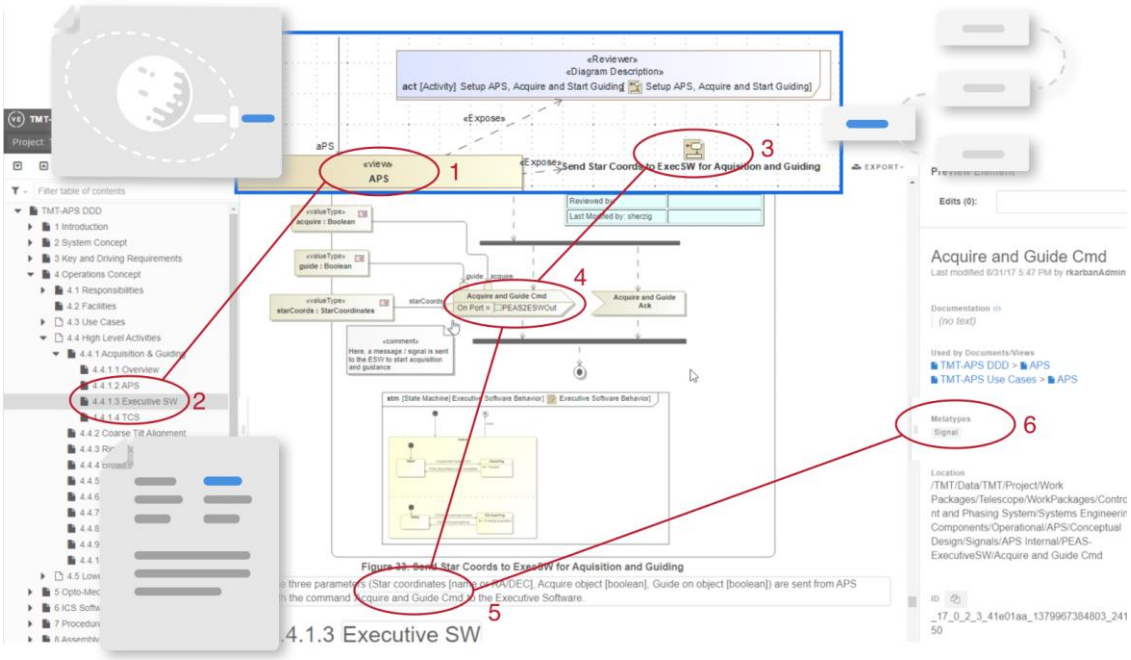


Figure 9. Documents as structured data and transclusions

The numbered items in Figure 9 can be described as follows:

1. Model representation of the view “APS”
2. View Editor rendering of “APS” in context
3. Exposed model diagram, in this case a SysML Activity Diagram “Send Star Coords...”
4. Model element, signal “Acquire and Guide Cmd”, in the context of the rendered diagram
5. Transclusion of the signal in line with narrative
6. Specification of the signal

Upon the document generation, as described in Figure 9, users can view the model diagram and can continue the model hardening process. A user can either manually generate the document through the web application or setup a periodic call to an external web service. This allows all users to see the latest generation of the document structure, though the diagrams are currently static, and the current version of model information which is always in sync. The querying and documentation of the model will start to reflect which ideas have been captured, identified, and analyzed and which ideas remain conceptual.

4.3 Systems Development Pipeline

The software industry has established best practices regarding pipelines for continuous integration and deployment. A pipeline provides feedback on every change, early detection and prevention of defects, collaboration between team members, automated release processes, and incremental and iterative lifecycle development. The typical software pipeline iterates through Design, Develop, Test, and Operate. The pipeline itself is deliberately designed to be process agnostic and customizable because it has to

support varying processes at JPL but it can be used to support a process such as ISO15288. The process owners at JPL typically collaborate with the providers of the Systems Environment to have their processes supported, enforced and verified.

The JPL Systems Environment has adapted this model to systems engineering and identified the following analogous phases (Figure 10):

1. Collaborate/Capture
2. Requirements/Design
3. Analysis
4. Publication

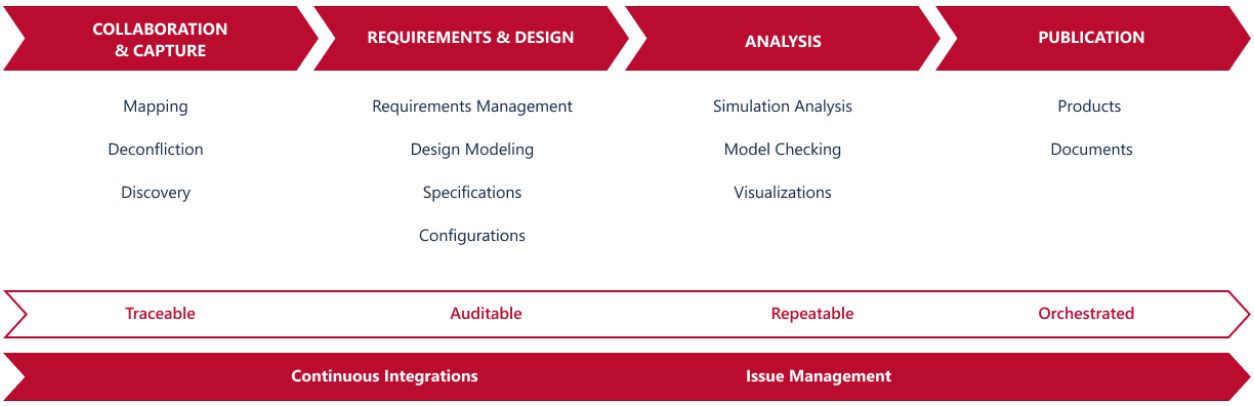


Figure 10. Systems Development Pipeline

Collaborate/Capture

The main focus in this phase is to discover, deconflict, and collaboratively map all relevant information. Different stakeholders contribute information about the system such as component properties, component behaviors, and mission scenarios. Initially, there is often conflicting or duplicated information which needs to be consolidated. For example, systems engineers would discover, identify system components in this phase. In a large system with many systems engineers collaborating on many components it is likely that there are duplicates which must be deconflicted or mapped.

Requirements/Design

During this phase, stakeholders leverage the identified properties, behaviors, and scenarios to write requirements, define architecture, and specify interfaces.
The components in the previous phase would be composed into an architecture with a description of their interfaces, specifications and configurations. Requirements would be developed for the components and their interfaces, as well as behavior and interactions specified; e.g. using state machines.

Analysis

The Analysis phase consists of behavior and structure analysis. Engineers analyze structure and behavior, define rules for model checking procedures, visualize the captured design in various views, and run simulations of different scenarios to perform trade studies.
A typical analysis is to simulate operational scenarios to verify timing requirements and use model

checking to check certain model properties such as reachable states in behavioral models. Query-based visualizations for certain views of structural or behavioral information is als a typical analysis.

Publication

Engineers integrate the products of the previous phases into documents by using OpenMBEE's transclusion mechanism as well as archiving snapshots of model-based documents.
Typical engineering documents such as requirements flowdown, functional design descriptions, or operational scenario analysis are automatically generated using the model data and combined with narrative.

The pipeline is implemented with a variety of applications and services and evolves perpetually as new capabilities are identified. The JPL Systems Environment supports an orchestrated pipeline for systems engineering tasks and products. These tools allow engineers to access data ubiquitously and transparently without having to rely on peer-to-peer integrations. Not all of the applications and services shown in Figure 11 are required, but they are provided and integrated according to the engineer's needs. The tools are a mix of commercial, open-source, and in-house built. Each of the mentioned tools serves one or more roles in the above pipeline.

4.4 Digital Threads

Digital threads preserve engineering work and provide traceability between documentation and models for impact analysis. A flexible web service architecture is designed to:

1. Be process and method agnostic
2. Bring your own application, data, and integrations
3. Deploy to the cloud with a continuous DevOps model
4. Enable multi-tenancy and security

Applications and data are integrated using a multitude of web services that address particular concerns and are selected for specific engineering use cases (see Case Studies). The web services conform to the OpenAPI specification [20] and provide an HTTP-based REST API for standard integration. Following the OpenAPI specification allows automatic generation of clients for a variety of languages. Different configuration managed digital threads can be built using this architecture. Figure 11 shows an example where one author creates an architectural model and stores it in a database, another engineer performs analysis and publishes results, and yet another engineer can aggregate information into a linked-data document using OpenMBEE's View Editor. In Figure 11, the red circles represent the end points of the digital thread.

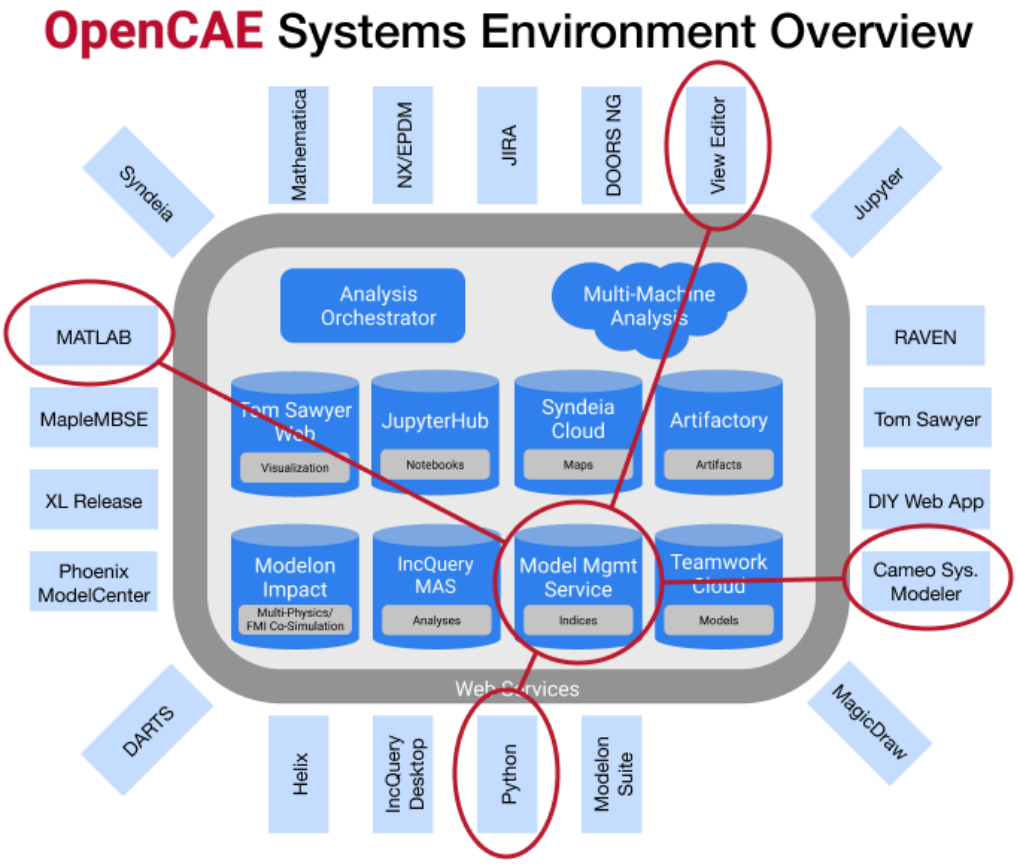


Figure 11. Systems Environment Architecture supports Digital threads

Digital threads can be constructed either by (1) explicitly connecting artifacts across different sources (e.g. DoorsNG [51], Artifactory [50], Jira [52]) which will form the total system graph and is maintained by Syndeia [19], or (2) establishing a pipeline process. Threads, at a minimum, provide traceability across the different types of data from different sources, e.g. requirements, model elements, artifacts. In a number of cases it is most convenient for engineers to interpret and apply the necessary semantics within the business logic operating on these relationships, for example a Jupyter notebook conducting a trade space analysis, publishing results with traces, and then collecting all previously generated results by trace for aggregated visualization.

Other use cases benefit from the integrating application possessing an understanding of the semantics of the integrated data. Syndeia, for example, supports model transformations between certain types of data from one source to another, e.g. synchronizing requirements from a requirements management service like DOORS with a SysML model in a modeling application like Cameo Systems Modeler. MapleMBSE [34] can project UML / SysML models to a tabular schema as defined by a declarative transformation language and enable collaboration on the same model through Excel, system modeling tools, OpenMBEE, etc.

Figure 12 shows an example of end-to-end utilization of the JPL Systems Environment Pipeline. A thread is traced through the Systems Environment Pipeline beginning with capturing data about system components using MapleMBSE [34] or Excel. MapleMBSE supports deconfliction enabling the systems engineers to work collaboratively on the same system model in their familiar tool which is Excel. Traditionally, engineers would modify (e.g. add, change, remove components) the Excel sheet locally and distribute it via email. With MapleMBSE they operate on the same (structured) data and see immediately the effects of the changes, or if something might conflict with something existing.
Then, interfaces are defined with Cameo Systems Modeler, visualizations are managed by Tom Sawyer, rules are checked by IncQuery [46], and documents are published to View Editor. The JPL Systems Environment enables integration and propagation for the user and avoids costly peer-to-peer interactions. OpenCAE refers to JPL's vision which provides an open portfolio in a shared environment that seamlessly connects engineers developing missions and systems.
Syndeia is only shown in "Collaboration & Capture" but effectively cuts across all phases as initially mapping and synchronization is established in the capture phase but then refined and elaborated in the others. Also, the pipeline process is iterative and incremental. Therefore, Collaboration and Capture is performed throughout the lifecycle.

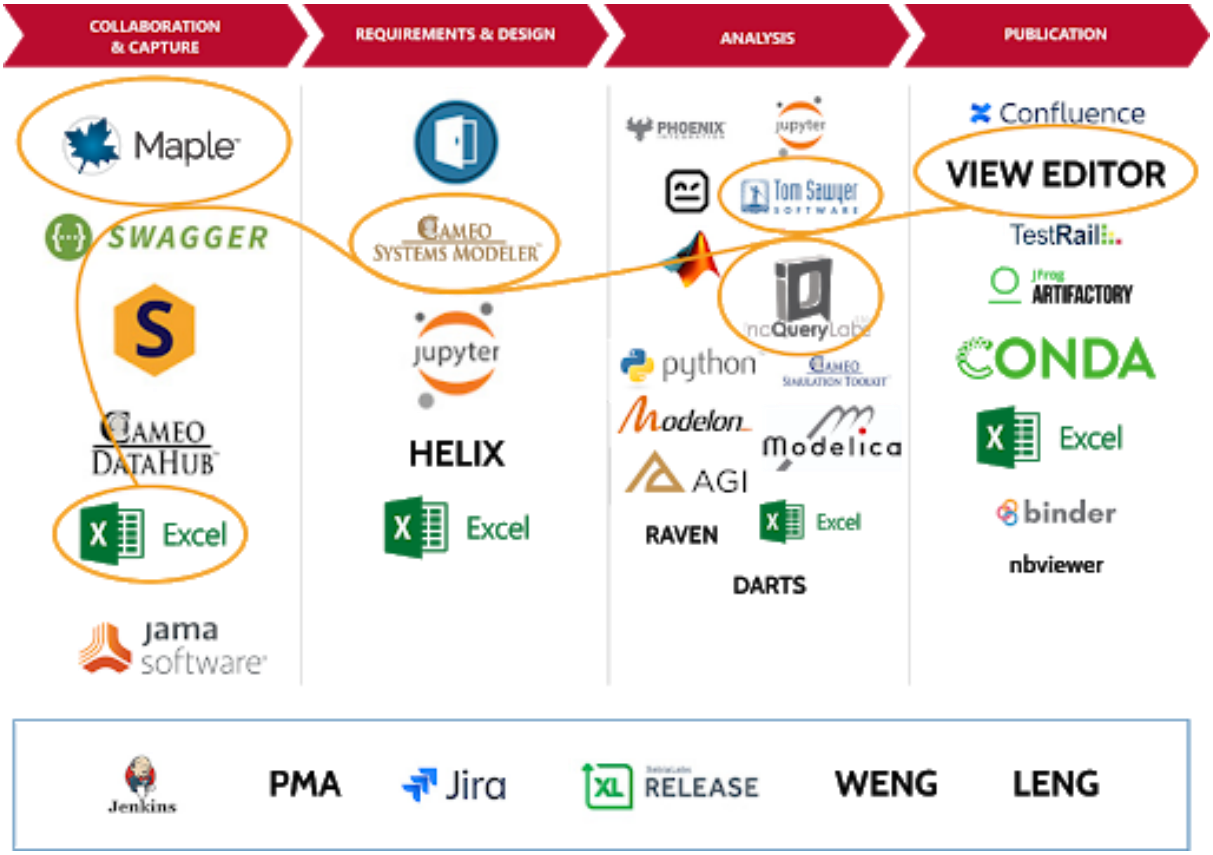


Figure 12. Digital Threads in the Systems Development Pipeline

The JPL Systems Environment is a continuously evolving system based around the OpenMBEE core in which new capabilities, services, and application are added as needed. Capabilities are integrated with the OpenMBEE core services as well as commercial solutions such as Intercax's Syndeia [19] which

configures the creation of controlled digital threads of engineering information. OpenMBEE plays the role of an enabler as it is not a standalone product by design. It requires integration with other tools and services to produce the linked-data documents and digital threads.

The capabilities of the JPL Systems Environment support JPL's 10 Systems Engineering (Table 1) functions which drive the JPL system development process. Existing capabilities are refined and new ones are added in perpetuity according to the projects' and engineers' needs. The capabilities are supported by the environment's applications and services infrastructure and associated workflows which explain how they interplay to provide the capability. Engineering Cookbooks capture the best system modeling practices and guide the systems engineer on how to efficiently use the infrastructure for a given capability.

	SE Function	Systems Environment Capability	Systems Environment Tools
1	Architecting	**Collaborative Workspace**, Data Model Design, System Modeling, Fault Protection, Design Communication, Capture Operational Scenarios, System Block Design, Visualize System Architecture	Cameo Systems Modeler, Cameo Concept Modeler, Tom Sawyer
2	Requirements	**Requirements Tracking and Management**, Requirements Design	DNG
3	Analyze and Characterize	**Capture Spacecraft Behavior**, Data Transformation, Large Dataset Analysis, Shared Computing Resources, Process Automation, **Multi-Domain Modeling & Simulation**, System Analysis, Model Checking Flight System and Software Behavior, Analysis Result Distribution	Systems Modeler, Tom Sawyer, Modelica, STK, Maple Workbook, Jupyter, Jenkins, PMA, Cameo Simulation Toolkit
4	Technical Resources	**Subsystem Data Aggregation**, Data Entry, Master Equipment List (MEL), Power Equipment List (PEL)	System Modeler, Maple Workbook, Cameo Simulation Toolkit, STK
5	Interfaces	Design Communication, Systems Integration Testing, Command Dictionary Management	DNG, Cameo Systems Modeler
6	V&V	System Testing, System Integration Testing	DNG, Helix, TestRail
7	Reviews	Collaborative Workspace, Role Capture, Track Model Analytics, Decision Traceability, **Document Generation**, Mission Operation System Coverage Reporting, Data Visualization	Smart Bear Collaborator, View Editor
8	Risk Management	**Connect Requirements to Engineering Artifacts**, Safety and Reliability Analysis	JIRA Risk Plugin
9	Change management	Process Orchestration, Issue Management, Connect Requirements to Engineering Artifacts, System Configuration Management, Analysis Parameter Tracking, Manage Engineering Change Requests	DNG, MMS, TWC, Syndeia, Artifactory
10	Task Planning	Process Orchestration	JIRA, XL Release

Table 1. Systems Environment capabilities support JPL Systems Engineering Functions

In order to facilitate an extensible environment capable of building digital threads, standard languages are key. These languages carry a significant amount of required semantics which are needed for digital threads. As a result, the number of languages supported by the JPL Systems Environment has been increasing as shown in Figure 13.

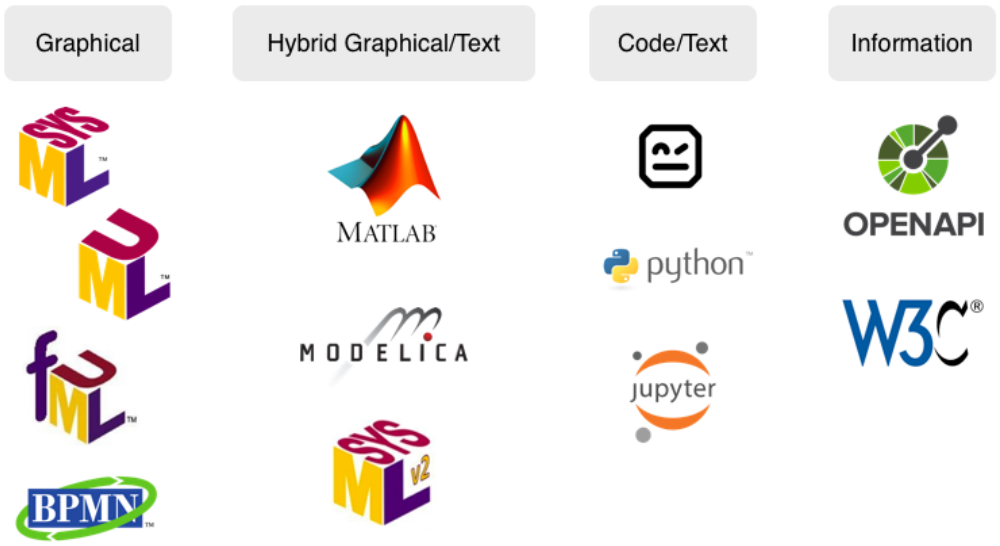


Figure 13. Systems Environment Supported Languages

4.5 Engineering Cookbooks

Engineering cookbooks are a compendium for the engineer which captures best practices, lessons learned, and provides guidance on how to use languages and tools to achieve a certain engineering task, such as "How-to Verify Requirements". The OpenSE Cookbook [22] is such a reference that is an open-sourced collection of patterns, procedures, and best practices targeted for systems engineers who seek guidance on applying model-based and executable systems engineering using SysML [7]. Its content has emerged from the system level modeling effort on the European Framework Program 6 (FP6) and the Thirty Meter Telescope (TMT) [16].

Application of the OpenSE Cookbook practices enables consistent delivery of engineered products using a well-defined modeling approach called the Executable Systems Engineering Method (ESEM) [18], which is a refinement of the Object-Oriented Systems Engineering Method (OOSEM) [23]. ESEM introduces the next phase of system modeling emphasizing executable models to enhance understanding, precision, and verification of requirements.

The OpenSE Cookbook provides a consistent, comprehensive, detailed, and background-agnostic set of operational procedures to guide practitioners through MBSE. Unlike existing SysML literature whose goal is to provide the foundations of descriptive modeling to newcomers and bring forward the arguments in favor of MBSE, the OpenSE Cookbook represents an implementation of what such literature often refers to as best practices or organization-specific procedures. It provides goal-oriented guidance for systems engineers explained by a set of combinable patterns. Systems engineering workflows drive each of the pattern definitions, such as how to verify requirements, roll-up technical resources, and analysis.

The OpenSE Cookbook demonstrates how to build and analyze system models using OpenMBEE [5] as applied to educational examples as well as actual usages in the TMT production model. While the JPL

Systems Environment, including its constituents like OpenMBEE and DocGen, is largely methodology and process agnostic, the OpenSE Cookbook provides field-tested patterns and examples that enable practitioners to shorten the time to productivity.

As shown in Figure 14, a TMT requirement change (1) propagates to the configuration managed SysML model where it is formalized into "property-based" requirements (2), allowing for a formalized trace of requirements into the design. A property-based requirement captures the quantifiable textual content (e.g. values, constraints) of a requirement as distinct SysML model elements that enable formal evaluation of said properties. The as-specified conceptual design (3) and/or the realization design (4) are verified against the changed requirement, providing immediate feedback of the change impact to the design, resulting in a pass or fail. The property-based requirement is bound in a parametric analysis context to the design to validate the requirement. The design information and associated analysis reports are delivered in various engineering documents (5).

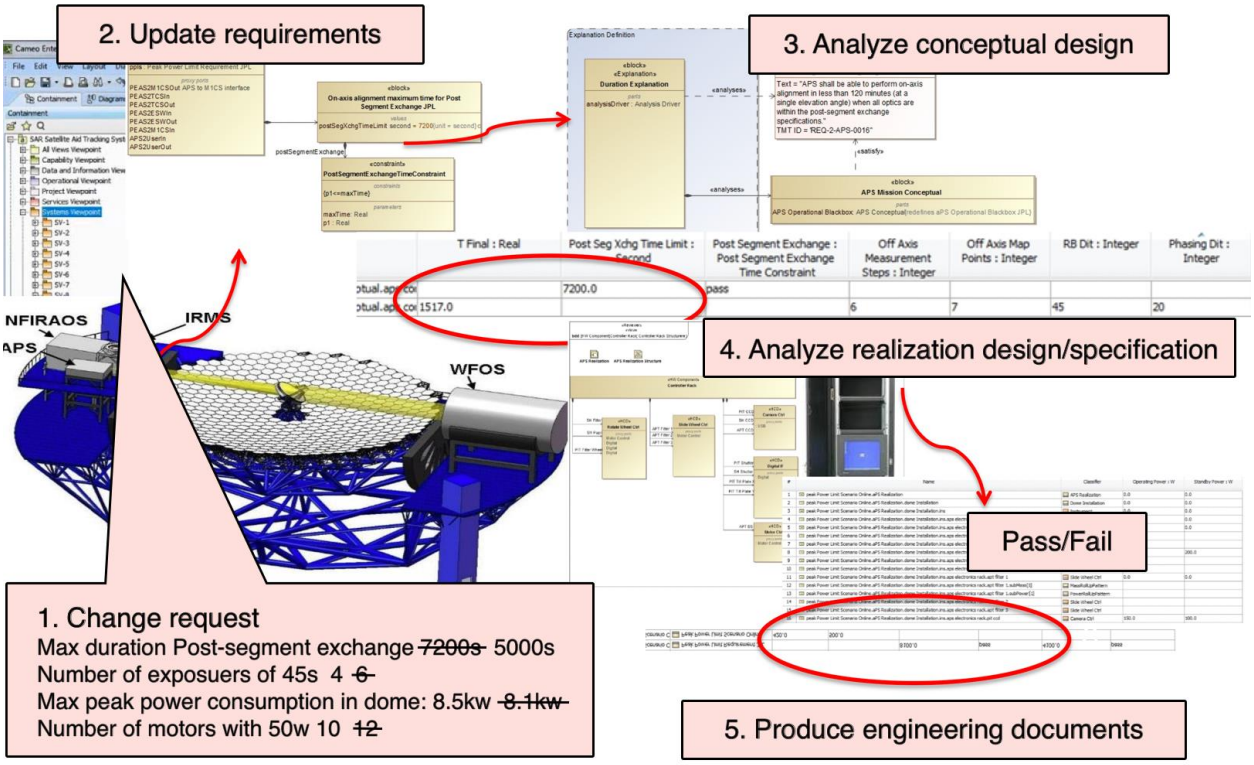


Figure 14. TMT analysis workflow

The overall goal of the OpenSE Cookbook in conjunction with OpenMBEE is to commoditize the Executable Systems Engineering Method, i.e. remove the cost and barriers to entry that allows for expanded innovation and broader operations driven by increased user access and decreased costs, in order to foster the broadest adoption.

The OpenSE Cookbook is accompanied by the OpenSE model library that provides model templates (e.g. to support ESEM), structural elements (e.g. for organizational charts), and behavioral elements to facilitate the authoring and analysis of models following the described patterns.

5 Case Studies

Since the OpenMBEE core elements have been open-sourced, there has been an ever-growing community using those elements to build their own Systems Environment. Table 2 gives an overview of JPL projects and Table 3 presents a sample of other organizations that have adopted OpenMBEE and presented their experience. The metrics demonstrate the scale of deployments which range from smaller research and development projects to full-scale flight projects. The Technical Readiness Level [24] varies, but the majority are at TRL8 which is defined as, “The technology has been tested and ‘flight qualified’ and it's ready for implementation into an already existing technology or technology system.”

Project	**Metrics**	**Presented**	**TRL**
JPL Mars 2020	PDR, CDR, 1M+ elements 90+ VE documents 10 projects 50 concurrent users	NAFEMS Oct 2019 [25]	TRL8
JPL Europa Clipper	PDR, CDR, 2.5M+ elements 230+ VE documents 4 projects 100 concurrent users	IS July, 2012 [27] IEEE AC Apr 2018[26]	TRL8
Thirty Meter Telescope	1 project, 5 users 300k elements	REFSQ 2018 [28]	TRL8

Table 2. JPL Projects That Have Adopted OpenMBEE

Project	**Metrics**	**Presented**	**TRL**

Boeing	40 programs, 1000 users	INCOSE IW 2020 [31]	TRL8
Lockheed Martin	50 projects, 100 users	INCOSE IW 2020 [30]	TRL8
Object Management Group	5 projects, 50 users	SysMLv2 Submission Team, SysML 1.5 RTF 2019 [29]	TRL8
Ford	4 projects, 10 users	INCOSE IW 2020	TRL7
JAXA	2 projects, 3 users	INCOSE IW 2020	TRL7
GTRI	6 projects, 20 users	INCOSE IW 2020	TRL7
GT/ASDL	5 projects	INCOSE IW 2020	TRL7

Table 3. Organizations That Have Adopted OpenMBEE

5.1 JPL Europa Lander

The Europa Lander is a proposed astrobiology mission designed to land on and sample the surface of Europa, a moon of Jupiter. The science goals of the mission concept would be to seek evidence of life on the moon, to assess the general habitability of Europa, and to further characterize the surface and subsurface to enable future robotic missions. The harsh Jovian radiation environment and heightened planetary protection considerations present unique systems engineering challenges for this concept.

From the full engineering infrastructure provided by the JPL Systems Environment, a subset was selected to reduce overhead and scope model development procedures. Time was allocated to develop the tooling configurations that supported project needs, such as configuration management, access control, and project-specific modeling patterns. The digital engineering architecture was designed such that there were multiple points of access to the system model for the flight system engineers to accommodate tool or workflow preference. A popular implementation of this principle is demonstrated by the use of MapleMBSE [34] to project the system model on to Excel workbooks by an isomorphic, reversible transformation where all cell changes result in created, updated, and/or deleted model elements to keep the system model consistent. The format of these MapleMBSE workbooks were derived from legacy Excel workbooks that maintained the system model without the rigorous backing supported by the JPL Systems Environment. All contributions made to the system model were version controlled and configuration managed by the OpenMBEE MMS in the JPL Systems Environment.

The JPL Systems Environment has the capability to manage the development of several Europa Lander flight systems engineering products:

- Master Equipment List (MEL)
- Compositional Systems Design
- Block Diagram

	A	C	D	E	I	J	K
1							
2		Subsystem	Component	Stage	CBE Unit Mass (kg	Unc %	Commonality
447		PWR-PYRO	CELL	LNDR	22.0800	5.00%	4: New
448		PWR-PYRO	CELLS	DS	9.2200	5.00%	1: Build-to-Print/
449		PWR-PYRO	CLOSE-OUT PNL	DS	0.4800	30.00%	3: Modified
450		PWR-PYRO	DBIS CARD	DS	2.6500	30.00%	3: Modified
451		PWR-PYRO	LBIS	LNDR	2.6500	30.00%	3: Modified
452		PWR-PYRO	MPS_0	LNDR	2.4700	30.00%	3: Modified
453		PWR-PYRO	MPS_1	DS	2.4700	30.00%	3: Modified
454		PWR-PYRO	PACKAGING	DS	2.3000	30.00%	4: New
455		PWR-PYRO	PDS	DS	2.4700	30.00%	4: New
456		PWR-PYRO	PFS_0	LNDR	2.4700	30.00%	4: New
457		PWR-PYRO	PFS_1	DS	2.4700	30.00%	4: New

Figure 15. Master Equipment List (MEL)

The MEL is a list of all component definitions used in the flight system design (Figure 15). These definitions include attributes that specify mass properties, subsystem categorization, and points of contact as well as other project-defined qualifiers. Mass properties from a CAD model were extracted using Syndeia connectors and maintained by MapleMBSE and View Editor.

Level 1	Level 2	Quanti	Type	Level 3
Europa Lander Vehicle				
Europa Lander Vehicle	BIO-BARRIER ASSY	1	BIO-BARRIER ASSY	
Europa Lander Vehicle	BIO-BARRIER ASSY	1	BIO-BARRIER ASSY	ADAPTER, CRS
Europa Lander Vehicle	BIO-BARRIER ASSY	1	BIO-BARRIER ASSY	EJCTD, UPR SHELL
Europa Lander Vehicle	BIO-BARRIER ASSY	1	BIO-BARRIER ASSY	FIXED, LWR SHELL
Europa Lander Vehicle	BIO-BARRIER ASSY	1	BIO-BARRIER ASSY	FIXED, UPR SHELL
Europa Lander Vehicle	BIO-BARRIER ASSY	1	BIO-BARRIER ASSY	FSTNRS
Europa Lander Vehicle	BIO-BARRIER ASSY	1	BIO-BARRIER ASSY	MRMN CLAMP INSTL
Europa Lander Vehicle	DOV ASSY	1	DOV ASSY	
Europa Lander Vehicle	DOV ASSY	1	DOV ASSY	3/8"FSTNRS, SEP NUT (
Europa Lander Vehicle	DOV ASSY	1	DOV ASSY	CUP, CATCH (ON DOS)
Europa Lander Vehicle	DOV ASSY	1	DOV ASSY	DOS ASSY
Europa Lander Vehicle	DOV ASSY	1	DOV ASSY	DOS ASSY
Europa Lander Vehicle	DOV ASSY	1	DOV ASSY	DOS ASSY

Figure 16. Compositional System Design Component Tree

A Compositional Systems Design document is produced by defining hierarchical relationships between components in the MEL. The component tree (Figure 16) contains both physical and conceptual components that define the partitions used in failure analysis. Since the components were selected from the model-based MEL, updates to either the MEL or component tree were propagated for consistency. MapleMBSE, Cameo Systems Modeler, and View Editor were used to maintain the Composition Systems Design. Hierarchical structure was also extracted from the mechanical CAD model via Syndeia connectors.

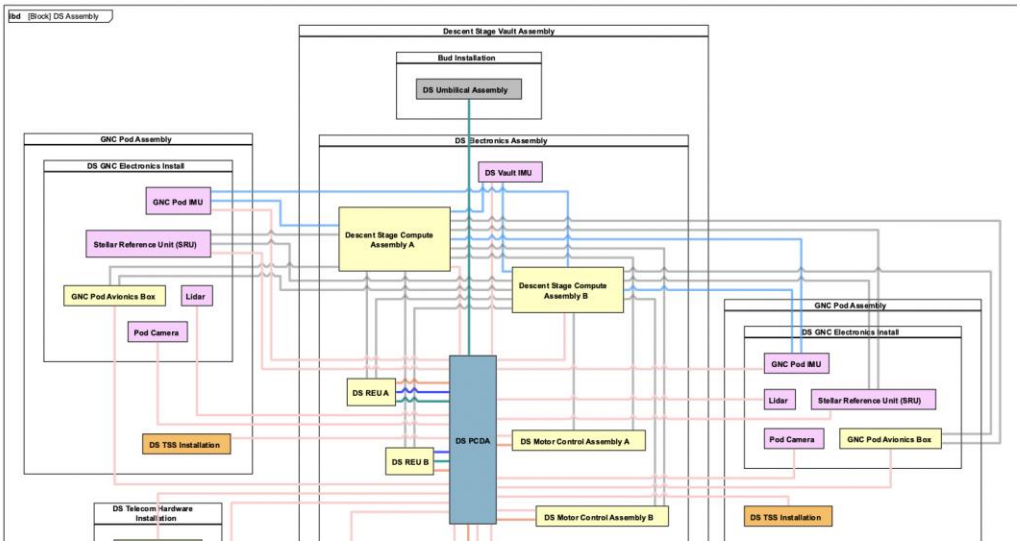


Figure 17. Flight System Block Diagram

The flight system Block Diagram is a graphical view of the flight system design. Components are represented as blocks with subcomponents within which follow the hierarchy defined in the Compositional System Design. In addition, the Block Diagram presents the power and data connections between components. This document was produced by a project-specific Tom Sawyer definition that rendered the Block Diagram from the system model data (Figure 17). The Tom Sawyer definition translates the system specification into interactive, color-coded blocks that represent the system composition and relationships between system model components. In this example, the project chose a visualization template emphasizing the connection between the components, eliding details of the interfaces. Those specifics are however captured in the model and typically represented as a table in the engineering documentation. Since the Block Diagram is derived from a central database, the hierarchy and component attributes were always in sync with the MEL and the Compositional Systems Design.

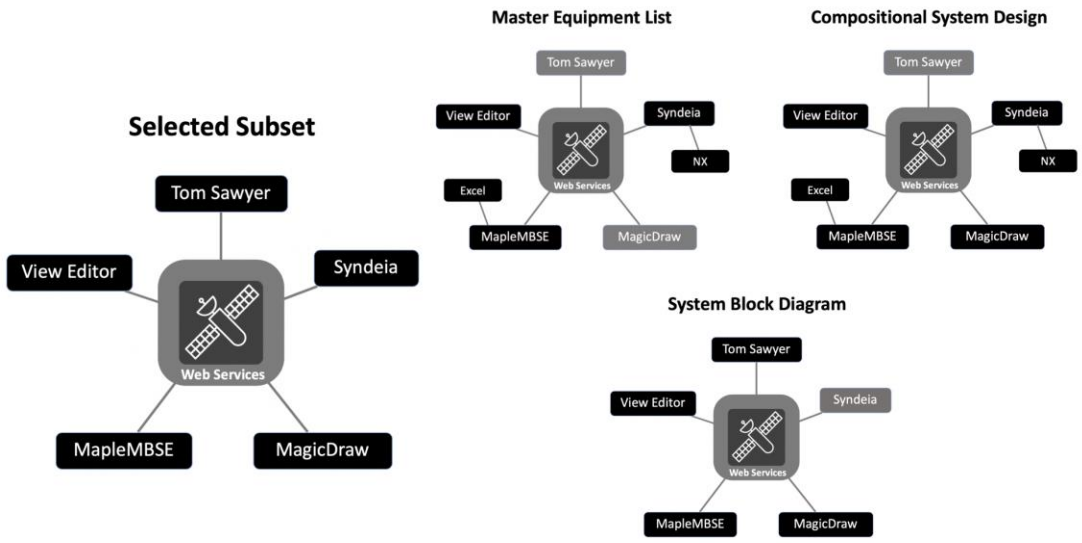


Figure 18. Subset of Systems Environment used for Europa Lander Concept

A different set of tools and services were utilized for each of the products in the Europa Lander Case Study as shown in Figure 18. For example, to produce the Compositional Systems Design, the relevant mass components are extracted from a CAD model with Syndeia [19], stored in a model repository service, queried by MapleMBSE [34], rolled-up in Excel, and published as report in View Editor. Syndeia provides a web-service and a user client. The connections between data sources are managed by the web-service but established in the client. The picture indicates that NX data is connected using the Syndeia client. The system environment, including OpenMBEE, is multi-tenant. So the same instance was used for all tasks but with different combinations of applications and services.

The Compositional Systems Design followed a breakdown structure to describe the assembled spacecraft. The environment tracked the rolled-up mass and reused common components as shown in Figure 18. The process followed the system development pipeline explained in the chapter, “System Development Pipeline”:

1. Collaborate/Capture
2. Requirements/Design
3. Analysis
4. Publication

5.2 JPL Mars 2020

The Mars 2020 rover, recently announced to be named “Perseverance”, will investigate a region of Mars where the ancient environment may have been favorable for microbial life, probing the Martian rocks for evidence of past life. Throughout its investigation, the rover will collect samples of soil and rock, and cache them on the surface for potential return to Earth by a future mission.

The JPL Systems Environment has been used extensively throughout the whole lifecycle of the mission which will launch in 2020 [25].

Two capabilities are demonstrated here:

1. Align Ground System & Operations Capability (Figure 19)
2. Generate integrated cross-domain documentation (Figure 20)

The Mission Operations System (MOS) operational scenarios were captured using BPMN models (top diagram in Figure 19). The Ground Data System (GDS) implemented software capabilities tracked in JIRA tickets. The MOS team tested each identified scenario in order to verify team roles and tasks as well as to test the GDS software. The JPL Systems Environment was used to trace and analyze the relationships between the identified scenarios and the implemented software capabilities. Once these relationships were captured, derivations were defined to identify which scenarios (the color differentiation of BPMN in top diagram of Figure 19) were supported by which GDS capabilities, thus creating a coverage matrix. DocGen was used to query the BPMN model and the reference model and generate a technical report (bottom diagram of Figure 19) .Using the OpenMBEE linked-data approach, the engineering work remained in sync with the calculated coverage and the produced documents. All three components contributed to a specified Integrated System Model (blue boxes in Figure 19)

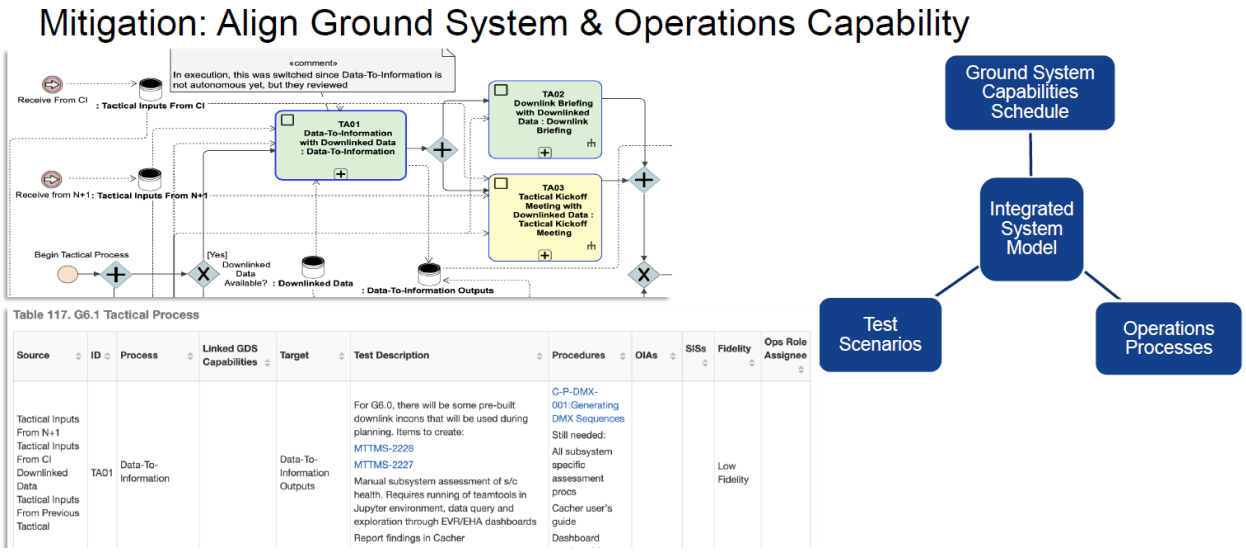


Figure 19. Align Ground Systems & Operations Capability

Similarly, another type of Integrated System Model was designed for cross-domain documentation. In this case, the electrical functions and reference designators were modeled in a standardized language, SysML, linked, and the produced document which consisted of many tables (examples seen in Figure 20) was generated through the mentioned OpenMBEE linked-data approach.

Over 90 linked-data View Editor documents that were collaboratively produced by approximately 50 concurrent users and resulted in 180,000 connections between documents and model elements.

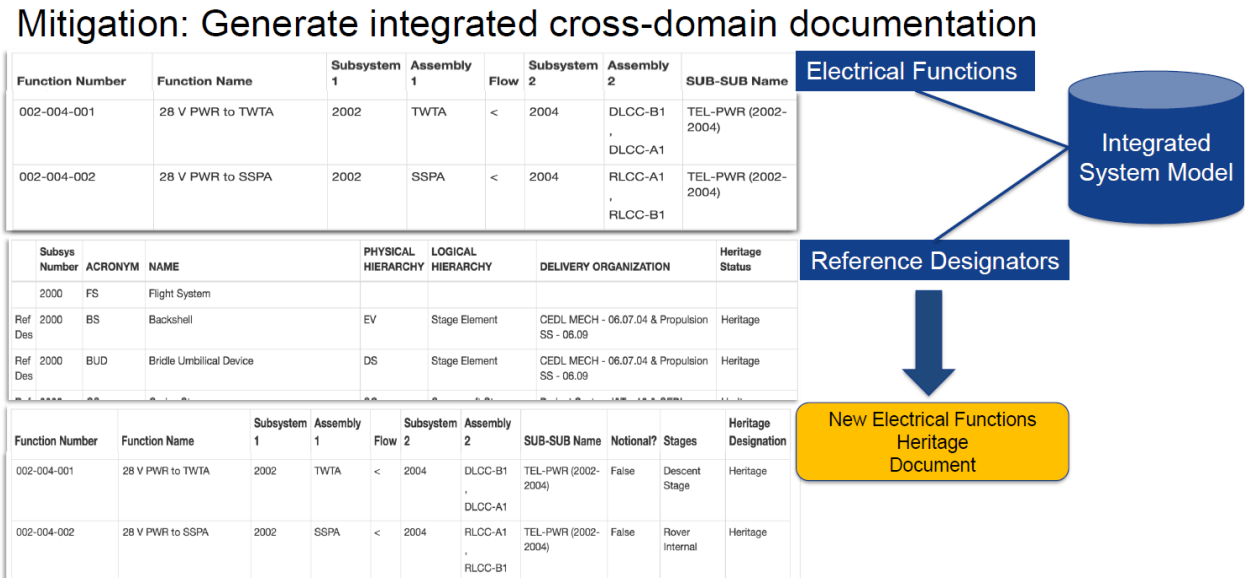


Figure 20. Generate integrates cross-domain documentation

5.3 JPL Europa Clipper

NASA's Europa Clipper will conduct detailed reconnaissance of Jupiter's moon, Europa, and investigate whether the icy moon could harbor conditions suitable for life. Europa Clipper has been the largest JPL project to utilize the JPL Systems Environment and OpenMBEE's linked-data mechanism [26][27]. Up to the Critical Design Review, the system model consisted of over 2.5 million elements and over 230 linked-data View Editor documents that were collaboratively produced by approximately 100 concurrent users and resulted in 445,000 connections between documents and model elements. Among those documents were Conceptual Descriptions, Requirements Documents, Block Diagrams and resource tables for mass, power, and data, as shown in Figure 21.

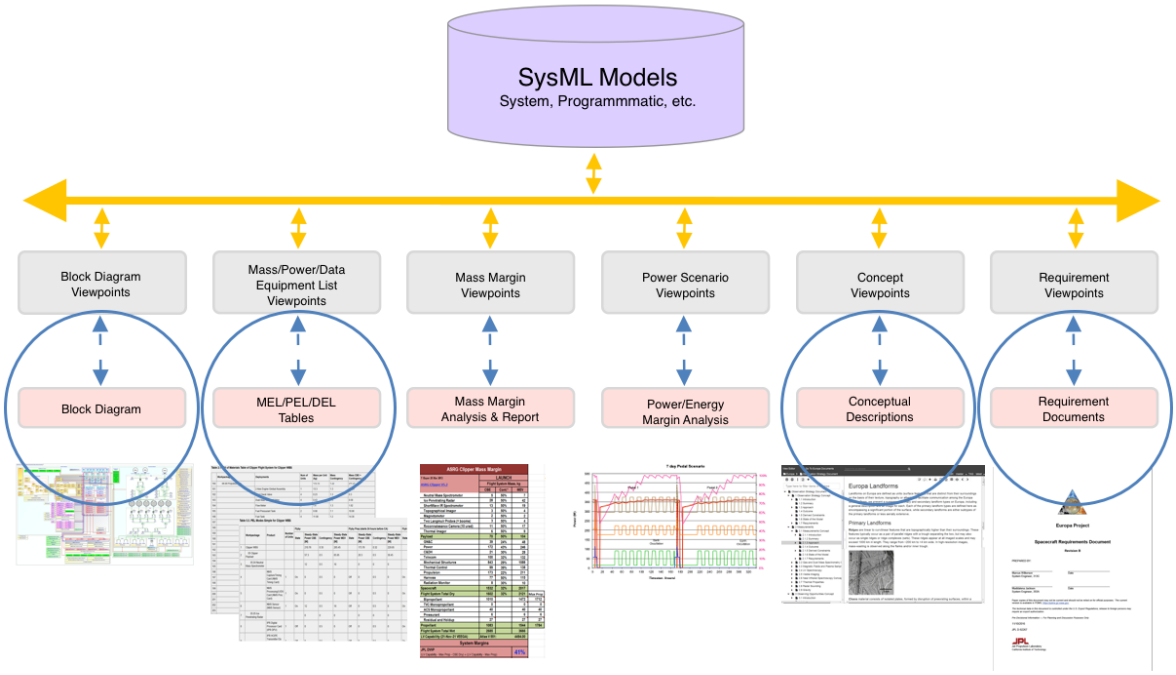


Figure 21. Digital products created by using the Systems Environment

The Europa Clipper Flight System Analysis formalized the resource management workflows, and published analysis results and timelines into the JPL Systems Environment services [32]. The process used HTTP-based REST API endpoints of services using clients generated from the OpenAPI [20] standard specification in different tools and languages, such as Mathematica, MATLAB, Python, and Java as shown in Figure 22.

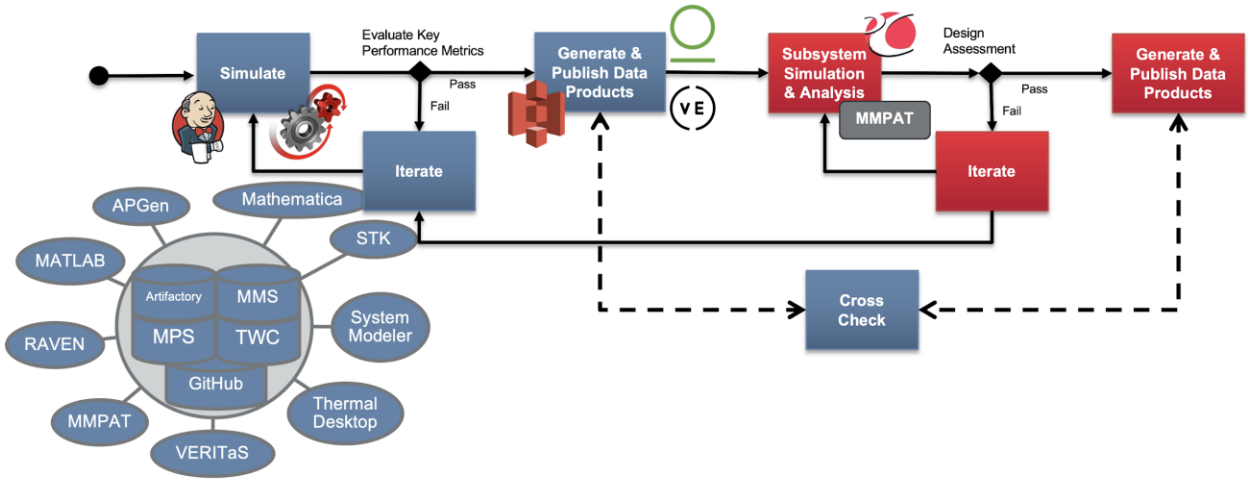


Figure 22. Europa Clipper Flight System Analysis Pipeline

5.4 JPL OpsRev

JPL's Operations Revitalization (OpsRev) project was one of the first projects to use OpenMBEE and an early version of the JPL Systems Environment. The project demonstrates how the use of DocGen managed documents significantly reduces the effort to produce consistent engineering documents. For the first review, the initial DocGen tooling was created by a systems engineer with less than 0.25 full time equivalent (FTE) allocation. The team then generated full documentation and review products, and turned around changes in less than 24 hours from the day of the review. The team developed ten substantial documents within a month with 3-4 FTEs. Tasks included systems engineering activities with approximately one FTE allocated to developing frameworks, tooling, and patterns. The managed linked-data document approach saved about five work-months of effort in one calendar month.

5.5 NAVAIR

NAVAIR used a surrogate pilot to validate their systems engineering transformation process using OpenMBEE's linked-data document approach [35]. OpenMBEE's View Editor was used for developing the technical description and design analysis reports (Figure 23), introducing digital approval and sign-off of model updates (Figure 24).

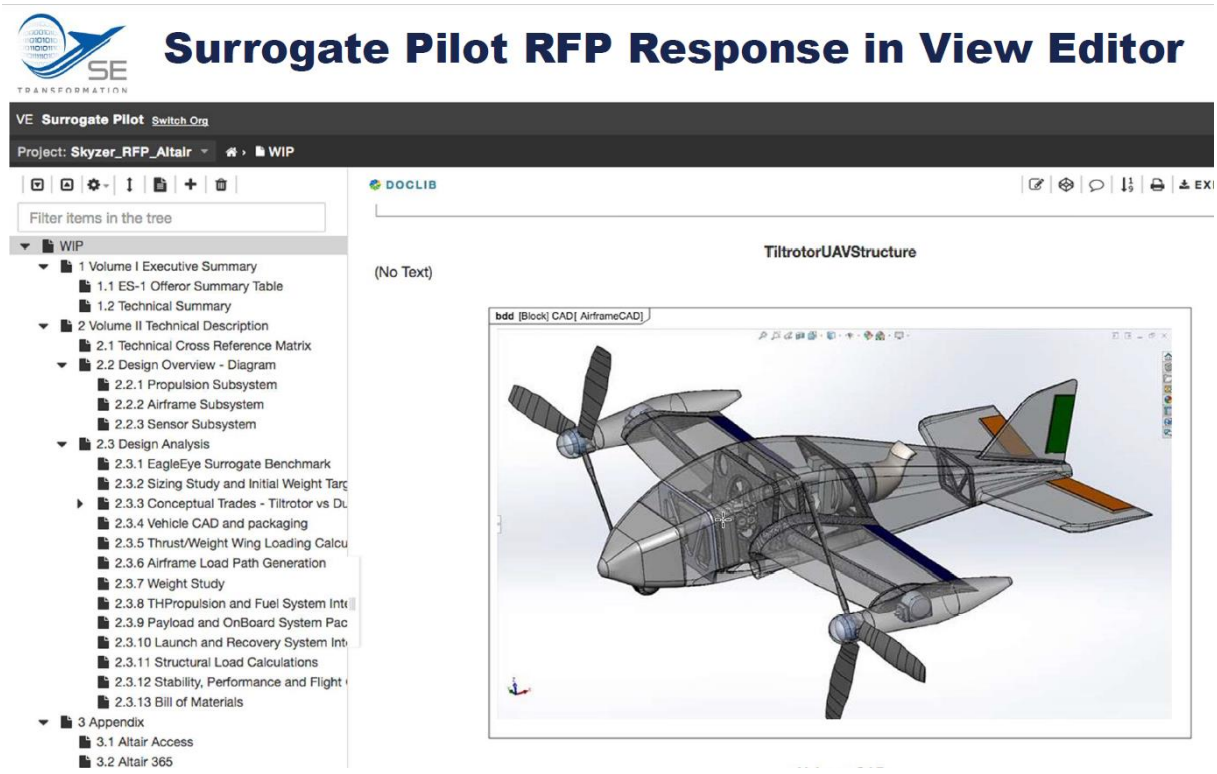


Figure 23. Technical description and design analysis reports

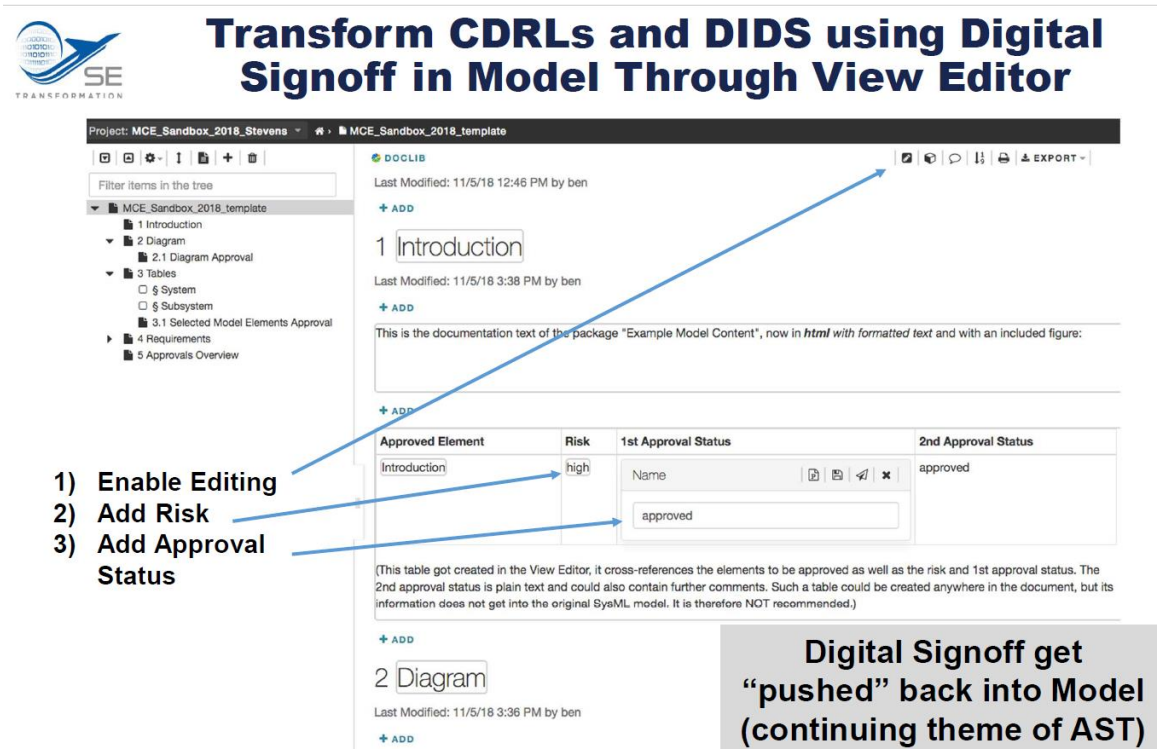


Figure 24. Digital Sign-Off

6 Results

The JPL Systems Environment aggregates different aspects to provide a powerful platform to support systems engineering activities including requirements management, design, and analysis. It is designed as an integrated set of best of breed web services to configuration manage the data and provide access to various clients for authoring, analysis and publication. The architecture enables building digital threads by connecting, propagating, and transforming data in a configuration managed way. The use of HTTP-based web services as a *lingua franca* across the environment simplifies the integration of new applications to accommodate human factors like preference and familiarity as well as an evolving technology landscape.

OpenMBEE is at its core with MMS and View Editor keeping engineering models and documentation consistent by using the transclusion mechanism. It de-references model data directly in the document text, and updates the information as it changes. The MDKs integrate MMS with a number of engineering applications such as MATLAB or SysML modeling applications.
The concepts of the systems development pipeline promote a flexible iterative and incremental process which can be adapted to the project's needs.
And finally, The OpenSE cookbook captures the best practices for Model Based Systems Engineering and demonstrates how the existing infrastructure is used to support the systems engineering tasks.

As shown throughout the paper, the research questions have all been answered satisfactorily.

How can models and documentation be kept mutually consistent and correspondent?

How can modeling languages be used collaboratively?

The linked-data mechanism, using transclusions to synthesize federated and multi-language engineering data into documents, keeps both the document and the engineering data mutually consistent and correspondent. The use of transclusions validates H2 since documents and model data virtually operate as a single unit. Since the synchronicity between the model and documentation is automatic, the user effort that is required to produce consistent engineering documents is drastically reduced, thus validating H1.

How can the live collaboration and connectivity paradigm work in the context of rigorous engineering information?

The web-based nature of View Editor and its View and Viewpoint paradigm enables engineers to work collaboratively. This approach allows users to mature unstructured data into structured data. It is not intended to replace, but rather supplement, other integration techniques like model transformation and is especially useful in earlier project phases when the data is typically more unstructured and lower fidelity. This quality is what satisfies H3, since the environment is flexible in terms of the 'model-hardness' that it tolerates during the ongoing maturing of a project. Additionally, the environment makes the data in models readily available to a wide range of clients that can use it for specialized analyses, including the application of other integration techniques.

How can engineering work be preserved and remain traceable to documentation for impact analysis to assess the consequences of changes in the model on documentation?

The transclusion mechanism ensures that engineering models and resulting documents remain connected and allows for model validation. Modern technologies, such as scalable query platforms (IncQuery Labs [46]), may be leveraged to check and validate the model and documents. Repository-wide change impact analysis assists in maintaining downstream transclusions by analyzing dependency chains across models and documents. MMS ensures that each change made to the system model and the document by an engineer is recorded in version control history. This validates H4, since the environment manages the configurations of documents and model data, i.e. which version of which model element is used in which version of a document. This allows to re-trace the source of authority for the resulting documents and any given time.

7 Discussion and Limitation

The OpenMBEE implementation scales well at enterprise level performance as demonstrated by case studies for managing tens of millions of model elements, hundreds of documents and thousands of concurrent users.

However, the implementation is currently limited by scalability of the current model hardening techniques. It is challenging to robustly incorporate modeling hardening through transclusion into tools that facilitate unstructured narrative, such as View Editor. References must be discovered and applied to efficiently update the set of transclusions for users without making mistakes for potentially hundreds of thousands of model elements. Currently, research is being carried out to automatically propose transclusions to the user, based on the contents of text documents, using natural language processing (NLP) techniques run on a dedicated graph representation of the model. If a text fragment could be a

reference to multiple model elements, the algorithm will prioritize the one that is the most strongly connected in the model to other matches in the text.

The collaboration experience is a challenge as it requires incorporating versioning and configuration management, which are not drivers for traditional live collaboration and connectivity approaches.

Further limitations are related to:

- **Software upgrades**
  Engineering software requires version control to integrate with the JPL Systems Environment in order to maintain configuration management across all created artifacts. The applications and services must be able to authenticate, authorize, and connect to web services to satisfy multi-tenancy and cyber-security requirements.
- **Workflow**
  Changes in the engineering workflow are sometimes necessary as some current workflows cannot be replicated in a model-based environment. For example, each engineer might create a single presentation for their particular task and all presentations are aggregated together before a review. However, when leveraging the model-based environment, a change of workflow is required to adapt to the collaborative environment.
  In order to contribute to the system model without bloating the version history, workflows should accommodate grouped changes.
- **Management of black box system model subsections**
  Some workflows require pieces of a system model to be sectioned from other teams (e.g. for competitive contracts) which requires a well-organized model and managed dependencies as well as tooling support. Moving from documents to fully integrated models introduces new challenges for managing quarantines of system model subsections because of the graph nature of the model data.
- **Project management structure**
  Siloed projects will have trouble maintaining an integrated framework and are forced to rework their interface of how data is propagated and synchronized. Disparate groups must settle on definitions of values and components. If this does not or cannot happen, then the value of the integrated model-based environment will be limited.

8 Related work

As mentioned earlier, the environment promotes a best of breed integration at environment level and not point to point to allow that all applications and services can integrate on our terms and not on vendor terms as it is often enforced by monolithic platforms. To the best of our knowledge, no platform provides a generic transclusion mechanism to ensure that engineering models and documents are mutually consistent.

8.1 ValiSpace

ValiSpace[38] is a collaboration platform where users calculate and store engineering data. The tool captures components and their attributes and performs analysis such as mass rollups on this data. ValiSpace is a web platform and also provides integrated requirements management. Reports, engineering budgets and charts are also stored.

8.2 Capella

Capella [37] is an open-source solution for Model Based Systems Engineering (MBSE). This solution provides a process and tooling for graphical modeling of systems, hardware, or software architectures in accordance with the principles and recommendations defined by the Arcadia method. Capella is an initiative of PolarSys, one of several Eclipse Foundation working groups.

Capella provides its own metamodel that defines the language concepts the user can enable in a Capella project. The user creates an instance of this metamodel and can then view the model from various perspectives through diagrams, aligning with the View/Viewpoint paradigm.

Recently, an MMS integration was built which allows storing Capella models in the OpenMBEE.

8.3 ModelBus

ModelBus [36] is a framework for managing complex development processes and integrating heterogeneous tools. ModelBus allows integration of tools from various vendors serving a variety of purposes. This integration creates a virtual bus-like tool environment, where data can be seamlessly exchanged between tools. Manual export and import of tool specific data is avoided, which is usually accompanied by manually executed data alignment steps. The data can be linked by establishing traceability.

The key concept of ModelBus for interoperability is the virtual bus-like service-oriented architecture and the way it processes the data transmitted via this bus. ModelBus can work on traditional artifacts like source code or binaries, but its full potential lies in the handling of models.

8.4 Dassault Systemes NoMagic Collaborator

Cameo Collaborator for Teamwork Cloud [39] is a web-based product designed to present models in a simplified form for stakeholders, sponsors, customers, and engineering teams. The tool allows editing and reviewing models.

The OpenMBEE DocGen language [1] has been integrated into the commercial Cameo Publisher software. Cameo Collaborator implements some of View Editor’s capabilities on a commercial platform.

8.5 Project Jupyter

The Jupyter Notebook and JupyterLab [13] web applications are integrated with OpenMBEE MMS allowing users to store, share, and collaboratively build executable Jupyter documents, or notebooks, in a wide range of programming and modeling languages.

8.6 Tom Sawyer

Tom Sawyer [45] is a commercial graph and data visualization software integrated with View Editor. View Editor can leverage Tom Sawyer to provide dynamic, interactive diagrams with advanced features like semantic zoom, smart layout, and graph algorithm execution.

8.7 IncQuery Labs

IncQuery Labs [46] provides an integration with OpenMBEE's Model Management System which allows model checking as well as validation of repository data using expressions formulated in SPARQL [47], VQL [48], and Lucene [49].

8.8 Jama Software

Jama Software [55] is a web-based, commercial, and collaborative mission engineering platform. Most commonly, Jama Software is used for defining, aligning and developing complex products, systems, and software for requirements and management of tests and risks. The platform leverages traceability to identify "suspect links" based on configurable relationship rules, and capturing and comparing baselined information during the development process.

9 Conclusions and Future Work

Managing complexity of projects is an increasing concern across systems and interdisciplinary engineering processes. It is of utmost importance to trace key decisions throughout the engineering process and maintain consistency across all artifacts. The JPL Systems Environment provides an open portfolio in a shared environment that seamlessly connects engineers developing missions and systems. The JPL Systems Environment is:

- **Open** - Processes, code, apps, services and artifacts are accessible by users as well as vendors and partners
- **Shared** - The diverse community of users, developers, partners and vendors are able to contribute
- **Connected** - Collaboratively construct and analyze the same precision products needed to develop Missions and Systems at JPL.

The integration of the JPL Systems Environment Pipeline across languages, tools, services, and repositories supports data synchronization, data propagation, single source of authority, provenance, configuration management, and traceability.

The upcoming SysMLv2 standard [29], which will also include a specification of an API for a system engineering repository, will drive the development of future OpenMBEE architectural versions.

The Jupyter [13] environment enables a new form of multi-paradigm modeling environment and brings with it a wealth of supported programming and modeling languages presented to users in an approachable way.

SysMLv2 and Jupyter will be key elements of future engineering modeling environments such as the JPL Systems Environment. Development on integrating Jupyter and the emerging SysMLv2 standard into the Systems Environment is already underway.

10 Acknowledgements

This research was carried out at the Jet Propulsion Laboratory (JPL), California Institute of Technology, under a contract with the National Aeronautics and Space Administration (NASA).

The TMT Project gratefully acknowledges the support of the TMT collaborating institutions. They are the California Institute of Technology, the University of California, the National Astronomical Observatory of Japan, the National Astronomical Observatories of China and their consortium partners, the Department of Science and Technology of India and their supported institutes, and the National Research Council of Canada. This work was supported as well by the Gordon and Betty Moore Foundation, the Canada Foundation for Innovation, the Ontario Ministry of Research and Innovation, the Natural Sciences and Engineering Research Council of Canada, the British Columbia Knowledge Development Fund, the Association of Canadian Universities for Research in Astronomy (ACURA) , the Association of Universities for Research in Astronomy (AURA), the U.S. National Science Foundation, the National Institutes of Natural Sciences of Japan, and the Department of Atomic Energy of India.

11 References

[1] DocGen, https://github.com/Open-MBEE/mdk/tree/support/3.x/src/main/dist/manual

[2] ISO/IEC, ISO/IEC 42010:2011, Systems and software engineering - Architecture description

[3] ISO/IEC, ISO/IEC 15288:2008(E), Systems and Software Engineering – System life cycle processes, International Organisation for Standardisation/International Electrotechnical Commission, February 1, 2008

[4] INCOSE “International Council on Systems Engineering.” http://www.incose.org

[5] OpenMBEE, https://openmbee.org

[6] W3C OWL, https://www.w3.org/OWL/

[7] OMG SysML, [Systems Modeling Language (SysML) Version 1.6], OMG, 2019

[8] OMG BPMN, https://www.omg.org/spec/BPMN/2.0/About-BPMN/

[9] Modelica, https://www.modelica.org/

[10] Zwemer, D., "Connecting SysML with PLM/ALM, CAD, Simulation, Requirements, and Project Management Tools", Intercax LLC, May 2011

[11] Mathematica, https://www.wolfram.com/mathematica/

[12] Python, https://www.python.org/

[13] https://www.jupyter.org

[14] Delp, C, Lam, D, Fosse, E., Lee, C., Model Based Document and Report Generation for Systems Engineering, IEEE AC, 2013

[15] Jackson, M., Delp C., Bindschadler, M., Sarrel M., Wollaeger, R., Lam, D., Dynamic Gate Product and Artifact Generation from System Models, IEEE AC, 2011

[16] TMT, "Thirty Meter Telescope." http://www.tmt.org

[17] Karban, R., Dekens, F., Herzig, S., Elaasar M., Jankevicius, N., "Creating systems engineering products with executable models in a model-based engineering environment", SPIE, Edinburgh, Scotland, 2016

[18] Karban, R., Jankevičius, N., Elaasar, M. "ESEM: Automated Systems Analysis using Executable SysML Modeling Patterns", INCOSE International Symposium (IS), Edinburgh, Scotland, 2016

[19] Syndeia, http://intercax.com/products/syndeia/

[20] OpenAPI/Swagger, https://swagger.io/docs/specification/about/

[21] Robot Framework, https://robotframework.org/

[22] Karban.R., The OpenSE Cookbook: A practical, recipe based collection of patterns, procedures, and best practices for executable systems engineering for the Thirty Meter Telescope, SPIE, Austin, TX, 2018

[23] Friedenthal S, Moore A., and Steiner R., [A Practical Guide to SysML 3rd Ed.] Morgan Kaufmann OMG Press, 2014

[24] Technical Readiness Level, https://www.nasa.gov/directorates/heo/scan/engineering/technology/txt_accordion1.html

[25] Fosse, E., From SysML to Mars, NAFEMS Model-Based Engineering, 2019

[26] Bayer, T., Is MBSE Helping? Measuring Value on Europa Clipper, IEEE AC, 2018

[27] Bayer, T., Early Formulation Model-Centric Engineering on NASA's Europa Mission Concept Study, 2012, INCOSE IS, Rome, Italy

[28] Karban, R. Requirements Analysis and Verification for the Thirty Meter Telescope with OpenMBEE and the OpenSE Cookbook, REFSQ, 2018, Germany

[29] SysMLv2, https://www.omg.org/cgi-bin/doc.cgi?ad/2017-12-2, https://omgsysml.org/SysML-2.htm

[30] Galey, C., Model Based Engineering Analysis, IW 2020 Torrance

[31] Przybylo, A. High performance graph query infrastructure for OpenMBEE, IW 2020 Torrance

[32] Brower, E. OpenCAE Case Study: Europa Lander Concept, IW 2019, Torrance

[33] Brower, E. Lee, E. JPL Systems Environment, IW 2020, Torrance

[34] MapleMBSE, https://www.maplesoft.com/products/maplembse/

[35] Cohen, D., SE Transformation - "Shaping our Future…", NAVAIR Public Release 2017-978, IW 2019, Torrance

[36] https://www.modelbus.org/

[37] https://www.eclipse.org/capella/

[38] https://www.valispace.com/

[39] https://www.nomagic.com/products/cameo-collaborator-for-teamwork-cloud

[40] Nichols, D., Lin, C., The Application of MBSE at JPL Through the Life Cycle, INCOSE IW, 2014

[41] MATLAB, https://www.mathworks.com/products/matlab.html

[42] JPL Mission Failure Causes 2019 - internal communication

[43] Europa Lander Study https://europa.nasa.gov/resources/58/europa-lander-study-2016-report/

[44] REST, https://en.wikipedia.org/wiki/Representational_state_transfer

[45] Tom Sawyer, https://www.tomsawyer.com/

[46] IncQuery Labs, https://incquery.io/

[47] SPARQL, https://www.w3.org/TR/rdf-sparql-query/

[48] VQL, https://www.eclipse.org/viatra/documentation/query-language.html

[49] Apache Lucene, https://lucene.apache.org/

[50] Artifactory, https://jfrog.com/artifactory/

[51] DoorsNG, https://jazz.net/products/rational-doors-next-generation/

[52] Jira, https://www.atlassian.com/software/jira

[53] Paredis, C. Model-Based Systems Engineering with SysML: Problem Definition, Analysis and Optimization, Model-Based Systems Engineering Colloquium, 2011. https://isr.umd.edu/event/6538/model-based-systems-engineering-colloquium-chris-paredis

[54] Mosterman, P. J., & Vangheluwe, H. (2004). Computer Automated Multi-Paradigm Modeling: An Introduction. *SIMULATION*, *80*(9), 433–450. https://doi.org/10.1177/0037549704050532

[55] Jama Software, https://www.jamasoftware.com/

[56] OMG SysML Extension for Physical Interaction and Signal Flow Simulation, https://www.omg.org/spec/SysPhS/1.0/PDF