Europa Clipper: MBSE Proving Ground

Todd Bayer, John Day, Emma Dodd, Laura Jones-Wilson, Andres Rivera,
Narek Shougarian, Sara Susca, David Wagner
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
Todd.J.Bayer@jpl.nasa.gov

***Abstract*—Recent collaborations between JPL's Integrated Model Centric Engineering (IMCE) initiative and the Europa Clipper project have produced six distinct applications of MBSE. Most of these collaborations have been successful, and all have been valuable learning experiences. Here we describe all of the applications including technical approach, benefits, challenges, and lessons learned.**

TABLE OF CONTENTS

1. INTRODUCTION

The future of systems engineering is undoubtedly model-based, but the road from here to there is neither clear nor straight.

Since 2010, JPL's Integrated Model Centric Engineering Initiative (IMCE) has led MBSE infusion at JPL. During that time, IMCE has had active collaborations in six areas with the Europa Project. While not all of those collaborations have been successful, all have been valuable learning experiences. This paper will discuss examples and lessons from each.

2. MBSE APPLICATIONS ON EUROPA CLIPPER

Table 1 shows a summary of all six of the MBSE applications, description, time in use and lessons.

Table 1. Summary of Applications

Application	***Description***	***In use***	***Key Lessons*** (+) positive or (-) negative
Mass Equipment List (MEL)	SysML/Magic Draw capture and rollup of component mass Web-based reporting via OpenMBEE	2011-present	(+) Start small: modest, incremental objectives (+) Produce familiar products with better methods (+) Involve end user continuously (+) Effective project/line collaboration is essential
Power Equipment List (PEL)	Add Power states/demands to MEL Provide static description to time-based mission simulation Web-based reporting via OpenMBEE	2012-present	(+) Leverage existing tools (+) Include only the necessary data in the model (-) Build documentation along the way (-) Integrate early with Electrical SE model
Power/Energy Simulation	Multiple tools chained to provide repeatable power demand, energy production,& battery state of charge profiles	2013-present	(+) Share time-based profiles with subsystems early on for detailed design (+) Run frequently to detect issues early (+) Validate against high-fidelity subsystem tools (-) Spend effort aligning system behaviors, PEL and subsystem tools
Architecture & Requirements	SysML/MD and View Editor (OpenMBEE) -based architecture and requirement development tools	Partial capability 2014; Retired unfinished in 2019	(-) Ambitious developments need risk management (-) Training the full team to think differently is hard (-) Tools and processes need to support end user (-) Parallel developments impede failure analysis

978-1-7281-7436-5/21/$31.00 ©2021 IEEE

Science Traceability and Alignment Framework (STAF)	Framework for tracing science measurement requirements to project, spacecraft and science instrument requirements Excel-based	2016 - present	(+) Simple but well-conceived tools can enable important conversations between engineers and scientists (+) Process of translating requirements to mathematical constraints helped rqmts validation
Electrical Systems Engineering	Eclipse/EMF-based authoring tool, git for CM, Leveraging open-source standards (OWL2-DL, SPARQL)	2019-present	(+) Continuous integration of SE products is possible (+) Familiar user interfaces lower barrier to entry (+/-) Rigorous approach highlights just how much of current ad-hoc processes need standardizing

Each of these applications was done at a different time. Leadership and implementing teams were different for each of the implementations, except MEL and PEL which were done by essentially the same teams. Some of the lessons may appear contradictory – which in part is explained by the fact that each lead and implementing team may have to learn the same lesson as the last team.

3. MASS EQUIPMENT LIST

The mass equipment list (MEL) implemented on Clipper consists of two parts based on two decompositions of the flight system. The first is called the Bill of Materials Mass Equipment List and the second is called the Deployment Mass Equipment List.

- Bill of Materials Mass Equipment List: This mass equipment list is based on the work package decomposition of the flight system. The masses of individual pieces of hardware are aggregated along lines of supply responsibility (work packages). The relationships between hardware and their supplying work packages were modeled as SysML dependencies with stereotypes indicating that they represented the project supplying something. The relationships between work packages and sub-work packages were modeled in the same way. Each piece of hardware in the mass equipment list has the Current Best Estimate mass expressing the estimate of mass based on known information, a contingency expressing how much the hardware could potentially grow as the design matures and the Maximum Expected Value expressing what the highest possible value of the hardware could grow to as the project matures.

- Deployment Mass Equipment List: This mass equipment list is based on the physical decomposition of the flight system. The masses of individual pieces of hardware are aggregated in assemblies which contain those pieces of hardware. The relationships between assemblies and the hardware that they contain were modeled via SysML contains relationships. This mass equipment list also has the hardware's Current Best Estimate mass, contingency and the Maximum Expected Value.

Both mass equipment lists are generated from an underlying SysML model via scripts that post processed the information in the model. The mass equipment lists are updated based on monthly inputs received from the instruments and subsystems with their latest mass estimates or engineering change requests (ECR).

Note that power and thermal information was also displayed in the power equipment lists according to both the Bill of Materials and the Deployment hierarchies.

The primary benefits of this approach were the following:

- Forced internal consistency due to single underlying model for physical and work-package based decomposition in the mass domains.

- It was quite general in that it allowed an arbitrary number of components in each subsystem or instrument to be described. This is particularly helpful as the spacecraft design changes over the lifetime of the mission. The general nature of the base model also allows for sub-work packages or assemblies to be created.

- The graph-based nature of SysML permitted the use of transformation techniques to export information. A web interface is used to publish the mass equipment lists and corresponding documentation allowing for any member of the project and external stakeholders to interact and view the mass equipment lists.

The primary challenges associated with this approach were the following:

- The underlying language and patterns used to describe the design (as for the PEL) were somewhat complex, resulting in custom post-processing scripts that were difficult to maintain and extend.

- The underlying model itself was difficult to update and the update process was full of pitfalls making training of new staff to curate it difficult.

- While combining more than one domain into a single model has significant benefits in terms of ensuring consistency, with each domain there are new requirements on the level of abstraction and the necessary decompositions of the system that are needed.

The lessons learned may be summarized as the following

- Having a single model which combines information necessary for thermal, power and energy analyses as well as information necessary for parts of mechanical analysis presents a challenge. Clipper is a large project and different practices, different levels of abstraction and different quantities of interest result in different views of the system in different groups. These differences can create the need for additional post-processing scripts or manual steps that make the outputs from the central model amenable for use in downstream analyses. To avoid the need for this additional "translation" layer one may choose to make the model easily extensible to different decompositions and naming conventions (that are mapped to each other in a consistent way). This however, presents an increase in the amount of information that needs to be maintained.

- The underlying language used to describe the system (SysML) and the MagicDraw/Cameo environment in which the model was created resulted in a relatively complex set of scripts to generate power rollups, mass rollups and thermal rollups. While the initial SysML patterns were simple, as more requirements were levied on the model, the patterns became more complex contributing to the complexity of the scripts. As a result, maintaining and extending the model was found to be very labor intensive and often error prone. Designing with maintainability, ease of use of tooling and extensibility in mind would certainly be one of the major lessons learnt.

This work was originally reported in [1], [2] and [3].

4. POWER EQUIPMENT LIST

The Power Equipment List within the MBSE framework used on Europa Clipper consisted broadly of 3 parts. These were as follows:

- Power States Section: a list of names of power states individual pieces of hardware could be in (for example Heater A, has Off and On states) captured in state machines and their associated Current Best Estimate power expressing what estimates of power are at any given time based on known information, a contingency expressing how much they could potentially grow as the design matures, and the Maximum Expected Values expressing what the highest possible value of the power values could grow to as the project matures. In addition to power information, the fraction of that power that would be converted into heat was also captured since in some instances due to storage of energy or radiation of energy away from the system via longer electromagnetic wavelengths, some of the power might not be converted into heat. In addition to power and heat dissipation information, the Power States section of the Power Equipment List captured information on where heat was dissipated. Due to Clipper's unique design, in some instances heat was reclaimed via a Heat Redistribution System consisting of a thermal fluid loop which transported heat from one part of the spacecraft to another. The fraction of heat dissipated to this loop was also captured.

- Power Scenario Section: This section mapped the states of individual pieces hardware (for example Heater A On) to the Power Scenarios of subsystems and instruments (Instrument A On). For example a power scenario would show that when Instrument A is in the On power scenario its component, Heater A, is in its On state also. Instruments and Subsystems were explicitly defined aggregations of many such components. The idea of using these aggregations served as a way of managing complexity by assigning a point of contact responsible for delivery of groups of hardware the ability and the responsibility to describe how they behave in a simplified way.

- Modes Section: This section mapped power scenarios to power modes. In most cases this was a 1-1 mapping but in some cases the mapping was 1 - many since these modes were also assigned data production behavior. For example: the Mode 1 and 2 of an instrument might be mapped to the same power scenario but and have the same power, but their production of data might be different. While power scenarios were defined for both subsystems/instruments and the flight system as a whole, modes in this context were only defined for subsystems and instruments. In the Clipper architecture, the flight system is described by a reduced number of system modes, where multiple system level power scenarios and hence subsystem modes configurations can occur.

The three power equipment lists are generated from an underlying SysML model via scripts that post processed the information in the model. The power equipment lists are updated based on monthly inputs received from the instruments and subsystems with their latest power estimates or engineering change requests (ECR).

The primary benefits of capturing power information in this way was that:

- It was quite general in that it allowed an arbitrary number of components in each subsystem or instrument to be described. This is particularly helpful as the spacecraft design changes over the lifetime of the mission. The general nature of the

base model also allows for sub-work packages or assemblies to be created.

- It managed complexity by mapping component states not to flight system modes (resulting in a very large number of permutations of states that would be difficult to understand) but to subsystem and instrument modes which were smaller in number, comprehensible and had single points of contact.

- The graph-based nature of SysML permitted the use of transformation techniques to export information. A web interface is used to publish the power equipment lists and corresponding documentation allowing for any member of the project and external stakeholders to interact and view the power equipment lists.

- The same model that was used to display MEL, PEL and TEL information was used to generate large parts of the time domain energy model and could be used as a source of independent validation for the energy model for snapshots of power consumption at different times.

The challenges of this approach were as follows:

- Subsystems and instrument aggregations were defined based on who was responsible for delivering them. This was very helpful in having a single responsible point of contact for behavior and power values for each subsystem. However, it had some drawbacks. When describing groups of components via modes, to minimize the number of modes needed, one should select aggregations which behave in a correlated way or in other words have a relatively small number of allowable permutations of states. Since the aggregations were based on responsibility rather than directly on this principle there were a handful of corner cases which would potentially need a very large number of modes to describe. For example, if we imagine that every subsystem had multiple heaters supplied by the Thermal subsystem and the instruments and subsystems were allowed to behave in a more or less independent way, the number of permutations of heater states would be large, resulting in a large number of modes.

- The way in which all of the above mentioned information was modeled relied on complex patterns in the SysML modeling language. The language was not well suited for describing simple tabular information, resulting in difficult to maintain scripts that were necessary to generate simple rollups of power.

This work was originally reported in [4].

5. Power/Energy Modeling & Simulation

Leveraging the design parameters and state database implemented in MagicDraw, collectively called the Europa Clipper Design Capture Model, a simulation architecture was developed to simulate the Europa Clipper power system and enable the project to quantify, on a continuous integration approach, key metrics such as: power and energy margins, impact of spacecraft design changes and impact of spacecraft

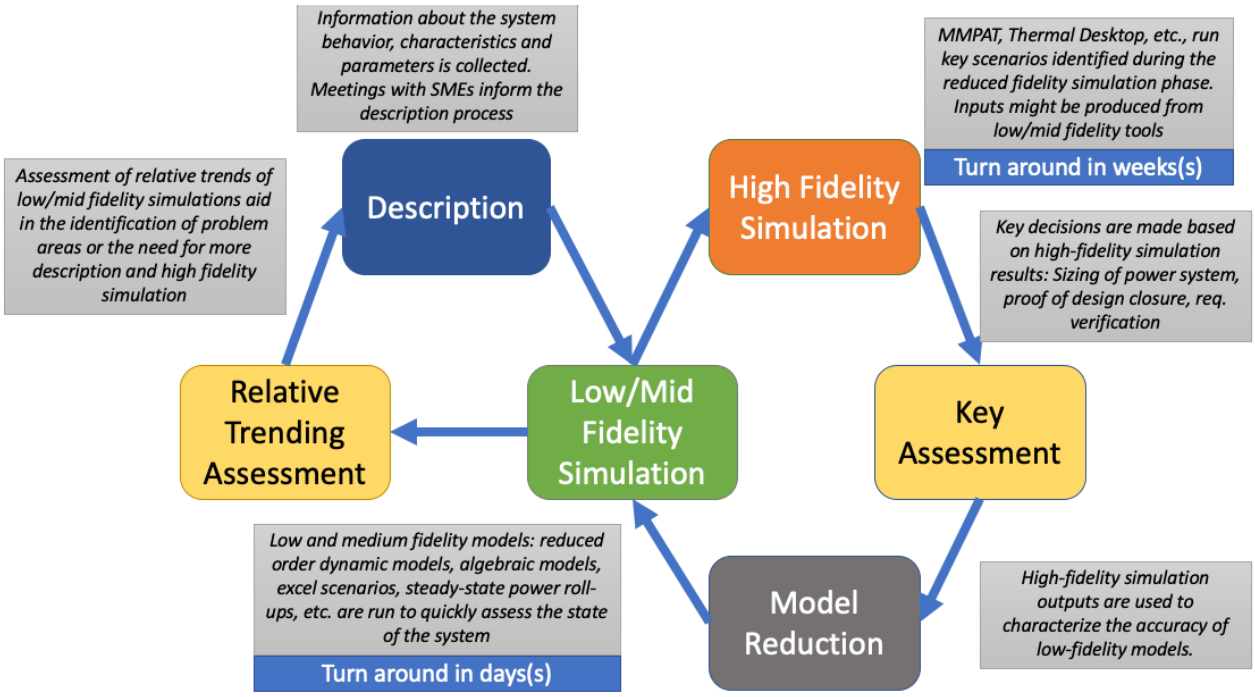


Figure 1. Rapid simulation approach

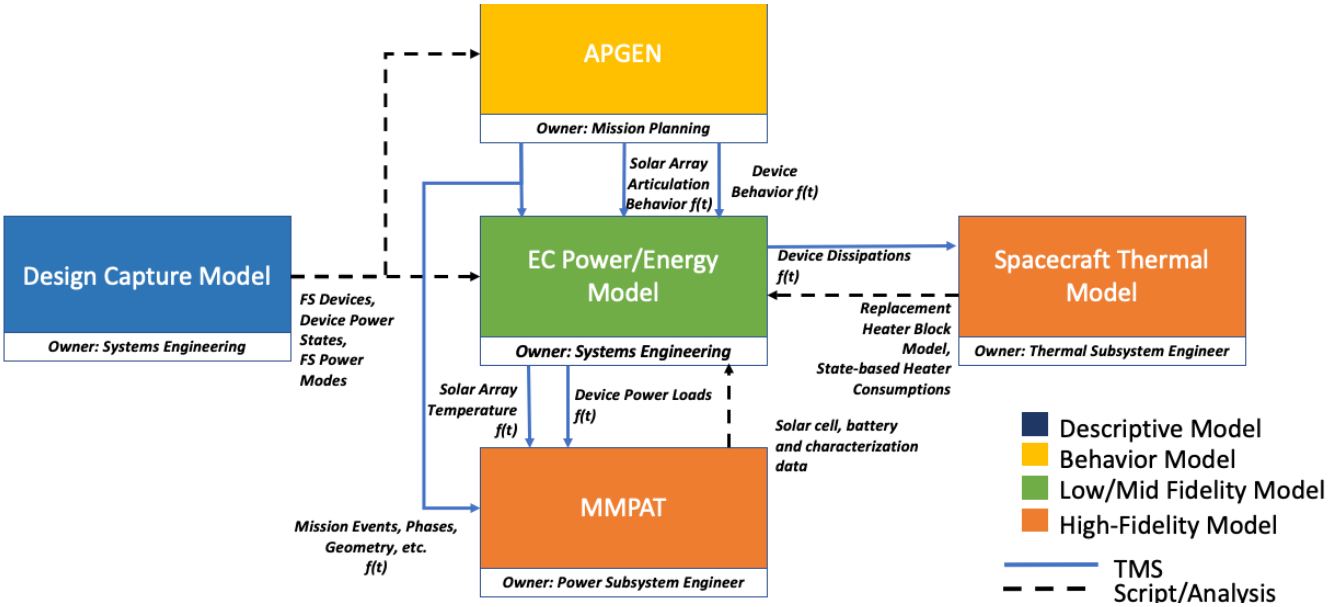


Figure 2. Primary models, interfaces and information flows

behavior and activity plan changes. Assessing these metrics is a problem that spans multiple domains and often requires the use of detailed high order models, implemented in tools that are not interoperable, and perform analysis in disparate time-scales, making a co-simulation approach not viable. The Europa Clipper Power/Energy Model solves this problem by taking a reduced order modeling approach and coupling the different domains into a single system level model capable of simulating the entire mission in less than 3 hours. Insight and products obtained from this model feed into the operating assumptions employed in higher fidelity simulation pipelines. A general description of the simulation approach and its use cases is described in Figure 1.

The Europa Clipper Power and Energy Model is primarily implemented in Modelica, using Wolfram System Modeler, but relies on multiple pipelines to transform detailed design information, test data or high-fidelity model results into reduced order models suitable for a multiple time-scale simulation. Modelica was chosen as the language of choice due to many of its inherent design features, such as the declarative mathematical syntax, making translation from mathematical notation to computer code a fairly easy process, its mixed discrete/continuous system modeling capabilities, allowing the modelers to express mixed behavior e.g. command driven state transitions triggering changes in continuous state quantities like solar array temperature predictions, and Wolfram SystemModeler was chosen due to its compatibility with industrial grade multi-time-scale DAE solvers, such as DASSL and CVODE and its integration with Wolfram Mathematica.

These models include:

- A descriptive model, the Design Capture Model, containing a semantic SysML graph representation of the Power Equipment List.
- A behavior model, APGEN, the mission planning activity plan generator and scheduling engine.
- A thermal model, the Spacecraft Thermal Model, providing a detailed geometric, mixed finite-difference/finite-element thermal model for transient and steady-state simulation.
- A high-fidelity power model, MMPAT, the Multi Mission Power Analysis Tool.

Scripted pipelines analyze, transform or perform automated code generation for data interchange. The flow of information is well documented, automatic scripts are orchestrated in Jenkins, a continuous integration platform. Time-based profiles for state variables or environmental inputs are published and read from a common time-series database, the Timeline Management System (TMS), while analysis results, analytical models and design characterization data are exchanged using the Inter-Office memorandum (IOM) process, the Engineering Change Request (ECR) process, or formally released documentation. A web interface is used to publish the resulting simulation time-series, allowing non-modelers and external stakeholders to access the simulation information. Figure 2 shows an overview of the flow of information between the different tools.

The taxonomy of the integrated Power/Energy model shows the coupled nature of the analysis domain, in architecting the model, power and thermal relationships are explicitly coupled, while environmental and geometric factors (sun range, radiation, attitude) are precomputed for a given trajectory and mission plan. The key elements of the model include the energy generation and storage devices: the solar array and the battery, a discrete power state load model, a dynamic replacement heater model based on the instantaneous power dissipations inside of the spacecraft vault, and a power distribution and power bus control model. In Figure 3, the flows of electrical quantities (current I, voltage V, and power P) are represented in yellow, the flows of thermal quantities (heat flux $\dot{Q}$) are represented in red, and

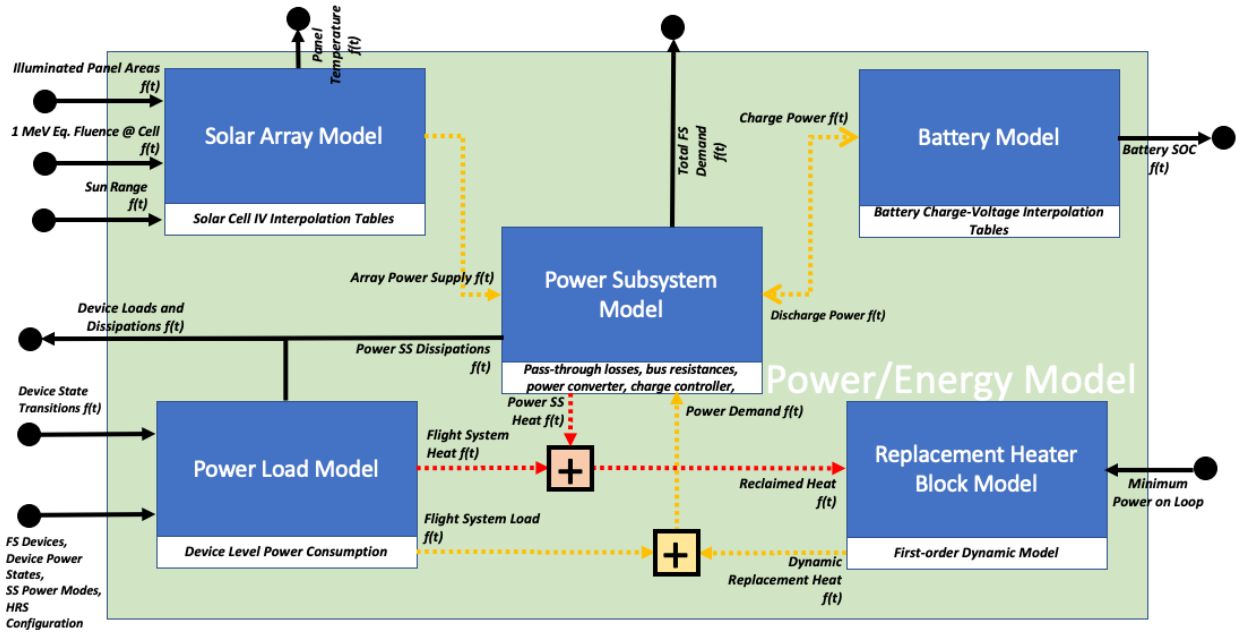


Figure 3. Power/Energy Model components and information flows

other state variable inputs and outputs are represented in black.

Power Load Model

Power loads are specified at the device level, where each state is mapped to a specific power consumption, this data is directly exported from the Design Capture Model by means of a restricted one-way transformation from SysML into predefined Modelica classes. A text file containing the index of the enumerated set of power states vs time drives the transitions of each power consuming element by means of a Modelica *CombiTimeTable*. The sum of power consumptions can be aggregated by their functional subsystem, or by their location in the physical decomposition tree; both mappings are directly exported from the Design Capture MEL/PEL model. Each element contains information about the percentage of power that is dissipated as heat into the Heat Reclamation System, allowing us to calculate heat fluxes across devices, subsystems or mechanical mounting location. A map of the relevant thermal parameters is maintained by the Thermal Subsystem team for accurate and detailed mapping of time-based heat dissipation profiles into inputs for detailed FEA/FEM analysis. The sum of all subsystem loads is mapped to the primary power bus, where the power consumption is converted into current and voltage, and power and energy computations are made. Figure 4, shows an excerpt of the component chain used to describe device loads.

Solar Array Model

The solar array model is based on characterization data for the Azur 3G28C IV solar cells, obtained through environmental testing at JPL data at various operating conditions spanning the environments experienced by the Europa Clipper Mission, including solar cell temperatures spanning the 100 C to -140 C rage, solar irradiances corresponding to spacecraft sun range of 0.65 AU to 5.46 AU, and radiation testing up to 5e15 1 MeV equivalent fluences. The test data is encapsulated as a 5-dimensional interpolation table, with axes corresponding to the 1 MeV equivalent

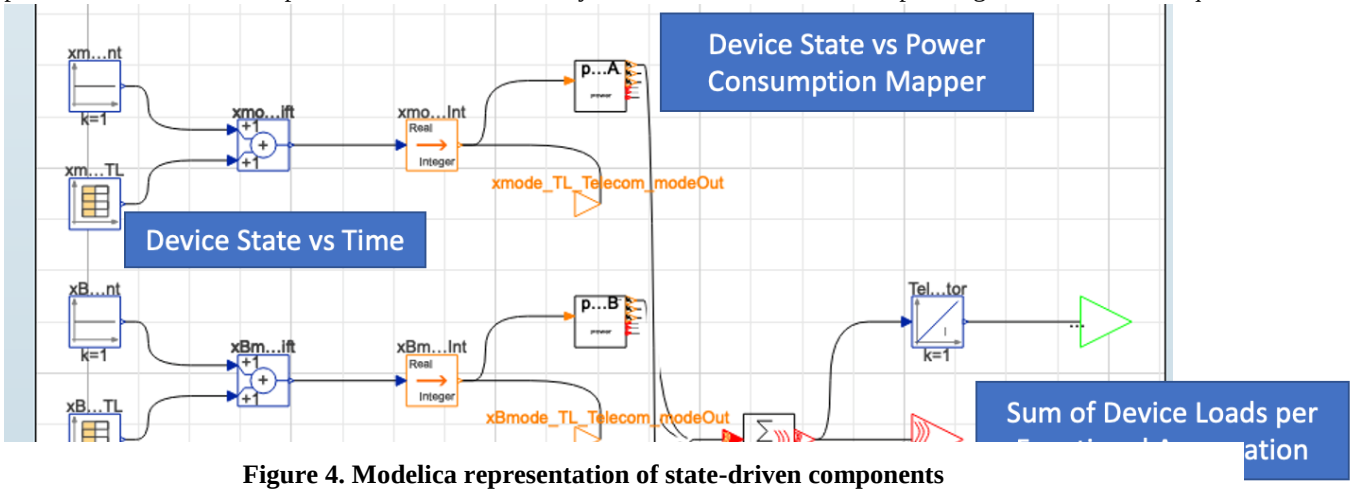


Figure 4. Modelica representation of state-driven components

electron fluence, solar cell temperature, solar intensity, voltage and current. The model uses this 5-dimensional table to trace the current-voltage (IV) curve of the system at any given period of time in the mission. The model can be run in a higher-fidelity mode where the IV curve is traced as the spacecraft demand changes dynamically or in a lower-fidelity mode, without a reduction in accuracy, where the maximum operating point of the solar array for a given environment and spacecraft configuration is predicted and the solar array demand is constrained. Environmental inputs for the solar array across the mission are distilled from higher fidelity models that generate time-based and attitude-based interpolation tables. These models have been implemented in AGI's Systems Toolkit (STK) in the form of plugins:

- The Jupiter Environment Toolkit (JET), which produces time series related to the radiation environment. This plugin depends on the relative damage coefficients for a given solar cell, as well as an input trajectory kernel that defines the position and velocity of the spacecraft at any point in time, producing a 1-MeV equivalent fluence time series for a given trajectory.
- The Solar Array Shadowing Lookup Table (SALT), which automates STK's built-in solar panel tool and performs a parameter sweep across 3 dimensions, the array articulation angle, the spacecraft solar longitude and the spacecraft solar latitude. The resulting table is queried by the model to obtain the shadowed area for any given panel, with a choice of including cosine losses as well as shadows cast by the spacecraft structure or other mechanical features. The output interpolation table is only dependent on a given spacecraft mechanical configuration, providing the flexibility to run multiple trajectories and attitude profiles without a costly shadowing analysis in the loop.

Battery Model

Europa Clipper's three 8s72p batteries are modeled using cell-level voltage vs capacity curves generated at different test conditions spanning the operating environment experienced during the mission. These curves are assembled in four-dimensional interpolation tables with the axes corresponding to the environmental inputs (temperature and loading condition), and the state variables, voltage and charge. The operating voltage of the battery defining the 0% and 100% state of charge points can be provided as an input, the model computes the capacity of the battery by querying the corresponding points in the temperature and load dependent voltage vs charge curve, while an integrator counts the load coming in or out of the battery to provide a dynamic state of charge prediction.

Power Subsystem Model

The power subsystem model implements a simple behavioral model of a DC to DC converter with fixed efficiency, where the bus voltage is provided by the battery and the solar array voltage is stepped down, serving as the primary power source. Resistances corresponding to the harness elements between the solar array and the input of the power converters, the battery and the primary power bus, and the power bus to the main power switches describe the distribution losses of the spacecraft. The power subsystem also implements the logic necessary to capture the charge management function of the power conversion and distribution assembly (PCDA).

Benefits

This simulation capability evolved over the years since its initial implementation in 2013 and along the way multiple benefits have been observed, including:

- Publishing results in a web interface increases team engagement, and consequently errors or incorrect assumptions can be questioned by a larger cohort of data consumers.
- Having a rapid simulation capability with sufficient accuracy to resolve most technical trades enabled the project to have a deeper understanding of the power and thermal domain without requiring lengthy simulation campaigns. The time saved in simulation time was used to review assumptions and results, improve the fidelity of the models and improve the capability of the toolchain.
- A project-owned power simulation toolchain allows independent validation of subsystem assessments for system level concerns. Subsystem analysis are typically targeted towards showing requirement compliance and often use overly conservative assumption. At the system level, some assessments require knowledge of the system capability and margin against the spacecraft as designed.
- The rapid simulation capability enabled more frequent reporting of metrics, increasing the visibility into the health of the system and allowing the project to trigger early margin recovery exercises when technical resources were predicted to be at risk.
- Modelica allows for the definition of general component libraries that can be reused in different missions.

Challenges

Designing a tightly integrated heterogeneous simulation effort poses challenges, primarily in the maintenance and synchronization of tools implemented in different platforms and languages. The primary challenges were as follows:

- The release of a particular version of the model was tied to a waterfall-like flow of releases from other tools. Sometimes the synchronization of the tool interfaces and the maintenance of the scripts posed additional delays in the release and closure of an analysis case.
- The representation of multi-domain concerns in a single model required the modeling team to receive inputs from different subsystems, sometimes these inputs were not consistent with one another or did not reflect the latest architecture implementation. In

some cases, incorrect assumptions were made due to the lack of relevant test or scenario data to inform the models.
- Some of the free open-source tools based on the Modelica language do not robustly implement the solvers required to simulate a multi-time-scale toolchain. Due to the complexity of our model, a commercial license is required to simulate an analysis case.

Lessons Learned

After 7 years of development, this simulation capability has changed quite significantly since its inception allowing us to identify the following lessons for the successful implementation of a multi-domain rapid simulation capability:

- Sharing of time-based profiles with subsystems as a unit of currency for the description of mission scenarios is key to the successful design of a system that meets system requirements. Continuous validation and checks of requirements and constraints against the different system profiles is essential to the understanding of state of the design.
- As the architecture evolves, the implementation state of the different models will diverge. A successful modeling effort should strive to cross-validate its low mid and high fidelity efforst on a regular basis.

This work was originally reported in [5].

6. ARCHITECTURE AND REQUIREMENTS

During every design effort, many competing factors are in play. The aim of architecture is to find and establish an achievable balance among them that ensures a satisfactory outcome for all concerned. However, interdependencies among the many aspects of a design can be extraordinarily intricate, especially when objectives are ambitious, but constraints are strict. Therefore, complete and precise communication is vital, both among developers and with those who define success. Establishing accepted value to stakeholders, unambiguously delineating responsibilities within a principled structure, making broad allowances for evolving implementation, and carefully articulating interfaces can assure the confident orderly progression of subsequent development and operation. This is the essence of good architecture.

Architecting is a process for developing and describing the characteristics of a system, such that it reasonably addresses the concerns of all stakeholders. The product of this effort is an architecture description that establishes the nature and organization of system components, along with principles and guidelines to govern their design, evolution, and operation over time. The basis for the architecture description was a defined Architecture Framework (AF) adapted/tailored from standards (esp. ISO 42010) and successful prior JPL practice.

The architecture framework used by Europa Clipper was intended as a guiding structure within which the usual development of a project could proceed, but in a model-centric manner, as opposed to the usual document-centric approach of longstanding practice [6]. It was not meant to be a distinct adjunct of that effort. However, as a first-time application of this approach, compromises were necessary and lessons have been learned that can be applied to future developments.

Summary Description

The Architecture Framework used on Europa Clipper established a set of elements and relationships from which the traditional engineering artifacts needed by a project could be composed. Within this relatively simple framework, most information is organized within a set of views, each dealing with some major aspect of the mission. The set of elements and relationships used to capture the architecture description are shown in Figure 5.

Conceptual views (*aka* concepts) are framed by topic or discipline, considering the overall system from some point of view. These are arranged hierarchically to address the flow down of **constraints** that originate as mission success criteria. These constraints are elaborated (e.g., through functional decomposition) and ultimately mapped to the design **elements** that comprise the system to be implemented and whose **relationships** determine their interactions.
Realizational views are framed by products. Conceptually developed constraints are assigned to the realizational **products** that must be developed, integrated, and deployed by the project, thereby attaining the status of **requirements**.
The architecture framework also provides for the documentation of **stakeholder concerns** in order to explain the motivation of mission **success criteria** and the processes for their establishment and accountability.
There are also provisions in the architecture framework for capturing key mission **scenarios**, the **analyses** that relate design and scenarios to mission success, various **models** that the project depends upon to conduct these analyses, and key **trades** that have determined the project's approach.

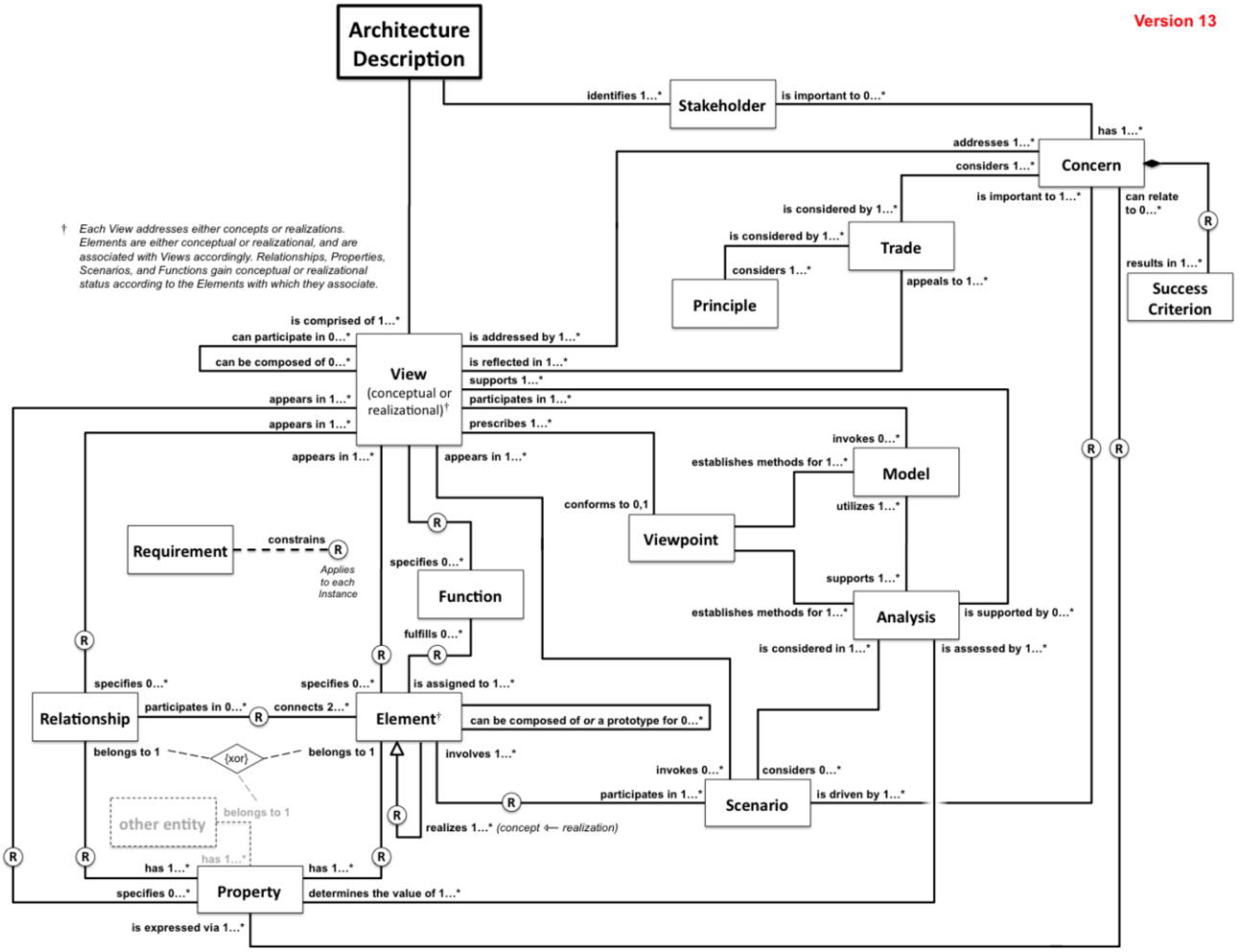


Figure 5. Elements and Relationships in the Europa Clipper Architecting Framework

Features

The system used to develop the architecture description on Europa Clipper had the following intended features:

- A web-based environment to enable users to collaborate on architecture definition and requirements development – model-centric, as opposed to document-centric
- An integrated database integrating all key elements of architecture description: stakeholders, concerns, scenarios, functions, elements, interfaces, requirements, trades, analyses, models, etc.
- Machine-computable ability to generate text-based requirements from mathematical constraints
- Ability to reconcile requirements taking into account multiple logical decompositions and physical limitations.
- Integration with other IMCE implementations

Implementation

When the architecture framework described above was devised and initially deployed on the Europa Clipper development, each of the framework categories was rendered within a relational database schema and deployed within a web-based content management system (CMS). This initial version, while responsive to users and requiring minimal training, was not used operationally, and the content was transferred to a common SysML-based repository to maintain consistency with other modeling initiatives on the project. The operational version was implemented using MagicDraw (SysML), and used View Editor (OpenMBEE) for document generation. A detailed description of the operational modeling approach can be found in reference [7].

While capable of capturing content at all levels of detail, to limit scope the architecture framework was applied only in the capture of Project and System-level architecture and requirements. This content was synchronized with the lower-level requirements captured in a DOORS-NG repository. Because of limitations in Magic Draw and View Editor search and query capabilities, the team developed an additional custom query/reporting tool. Content was added to the architecture description continued over the course of Phase AB development. During this time, project documents (gate products and requirements documents) were drawn

from the system model.

Benefits

The application of a framework for the Europa Clipper architectural description, and the tooling that supported it, had multiple benefits for the project team:

- Good description of stakeholders and concerns, and a meaningful trace to the project requirement set
- Better capture of full requirement rationale in narrative documents, capturing the rich interrelationships between design constraints
- Automated checking and reporting of requirements characteristics, part of a larger auditing framework made possible by the modeling infrastructure
- Generation of documents and other artifacts by query; enables fast turn-around and assures consistency
- Views tailored to the task; authoring views focused on input, whereas reporting and delivery views are focused on preferred content needs and preferences
- Cross-referencing of information to authoritative source; every fact and characteristic is captured in a single place, ensuring consistency when content appears in multiple documents

Additional information on benefits can be found in r*e*ference [2].

Challenges

The novelty and scope of the architecting framework was a barrier to its application. While a detailed framework definition document was available, it was not completed. The addition of tutorials helped, but in the end the deployment of a new systems engineering approach, in combination with unfamiliar and immature tooling, was too much to absorb while still conducting the business of the project. In particular:

- Ambitious approach involved developing methodology, tooling and training in parallel.
- Successful implementation required the entire engineering and management team to be retrained to think in AF terms – this proved to be impractical
- Incomplete tooling and training left users without the means to do the complete architecture definition, including especially requirements
- Requirements were developed using hybrid of new/traditional approaches. In the end they were late and were found to have significant issues with flowdown and leveling.
- ViewEditor editing/synchronization capability was not mature or scalable.

Lessons

After the project PDR, when a new leadership team came onboard, they found the partial implementation of the architecture framework too impractical to use and ended up reverting to traditional tools, terminology and methods. Understanding the causes for this inability to successfully deploy the architecting framework on Europa Clipper was difficult due to the highly convolved development. Given the lessons learned in this attempt, the following recommendations are offered for consideration:

- Avoid placing new methods and/or tools in the critical path of requirements development without effective risk management strategies.
- Early establishment of the adequacy and stability of tools deployed to support a project is fundamental.
- When deploying new methods and/or tools, ensure that the vision and objectives are shared across the management and engineering teams, to avoid working at cross- purposes. Significant changes can occur in an orderly fashion only when everyone is working to a common, shared goal
- Include ample and concrete examples in training material
- When generating the SE Management Plan, describe infrastructure and tools to be developed, and account for them in the schedule
- When deploying new methods and/or tools, plan for and manage personnel transitions, so that the value of the innovation is retained, despite potentially unfamiliar approaches.

7. SCIENCE TRACEABILITY AND ALIGNMENT FRAMEWORK

The Europa Clipper mission has a rich suite of ten science instruments, all of which must work together to collect data simultaneously in the resource-limited and hostile environment near Europa. The level of coordination required drove a need for broad understanding of the science across a large team of scientists and engineers, and the ability to integrate science value (and its partials) into the broader project systems engineering processes like requirements flow down and trade studies. So in an effort to implement a science systems engineering process equal to this challenge, the Science Traceability and Alignment Framework (STAF) was developed. STAF is an MBSE-inspired construct that centralizes and standardizes information about how instruments and measurement requirements contribute to a given science objective. The strength of this tool is the ability to map multi-dimensional data into a set of P-STAF and M-STAF matrices that are broadly accessible to both scientists and engineers, all while retaining much of the nuance that underpins the way the science is accomplished. This work was reported in [8], [9] and [10].

Features

The Science Traceability and Alignment Framework (STAF)

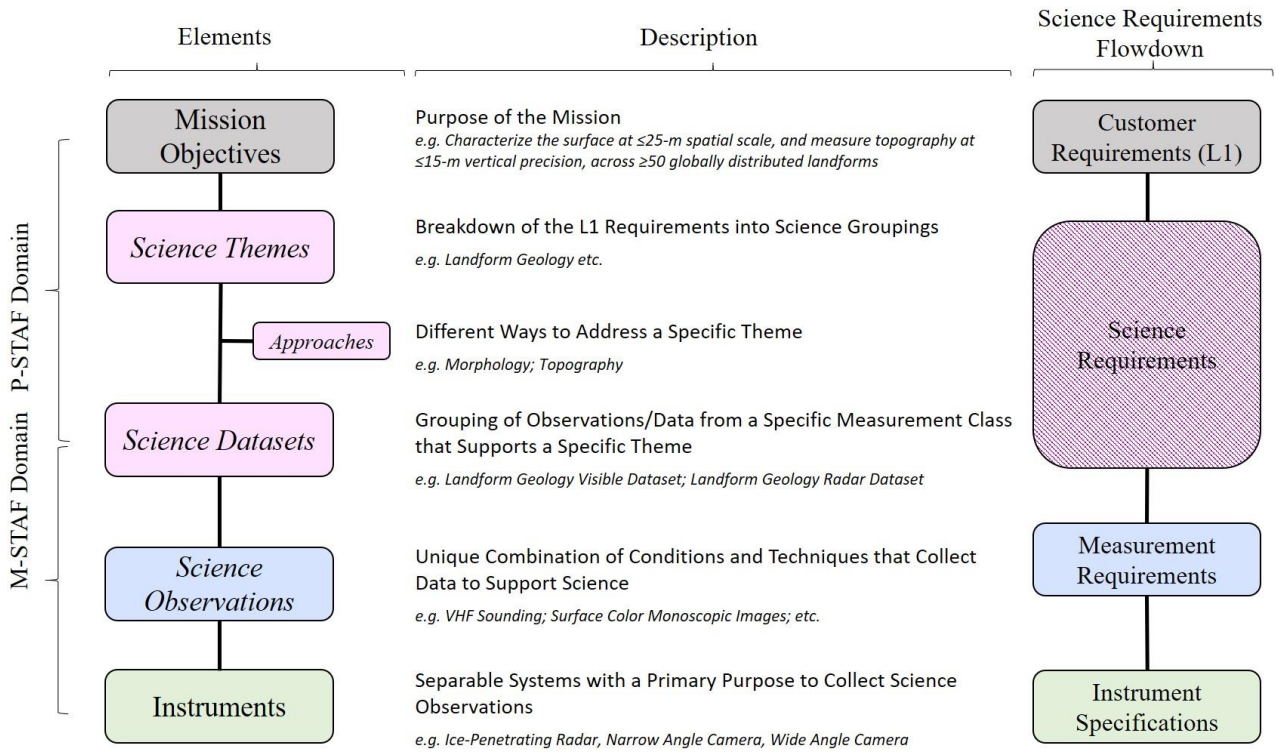


Figure 6. The taxonomy of the STAF elements

is a framework for tracing project science requirements to measurement requirements and eventually the instrument performance requirements. The basic taxonomy used on Clipper is described in Figure 6. Once the elements in the framework are populated, STAF provides a way to codify AND/OR relationships and relative strength of measurement contributions to science objectives.

The two main tools that leverage this framework are the Project-domain STAF (P-STAF) matrix and the Measurement-domain STAF (M-STAF) matrix, shown in Figure 7. The P-STAF matrix for the project expresses how a particular set of science measurements contributes to the L1 Mission requirements via science datasets. In particular, the columns of P-STAF are the various science observations and the rows are the L1 Mission requirements, expanded to describe the multiple ways each requirement can be achieved. Each populated cell is a science dataset that represents a contribution of the observation to the requirement. The power of P-STAF is the ability to identify and communicate different combinations of measurements that can meet a specific L1 requirement and their relative strength.

The set of M-STAF matrices (one for each science investigation on the mission) on the other hand defines a thorough and complete set of constraints necessary to execute a particular valid science dataset. Its rows correspond to the contributing science observations in the investigation's

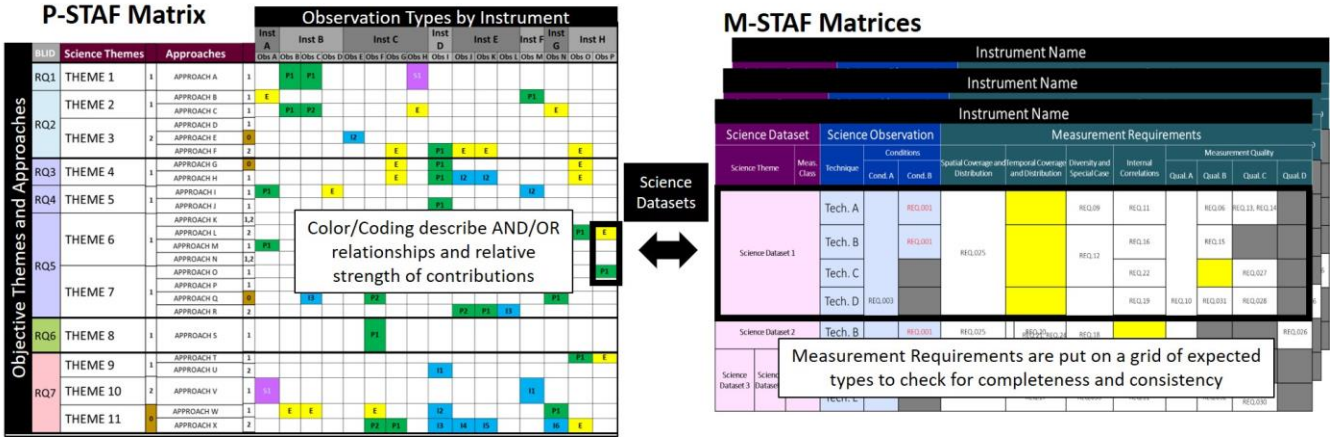


Figure 7. STAF Matrices and their relationship

datasets. The columns include different categories of constraints necessary to make the contribution valid, such as geometric conditions for the observation, required temporal and spatial coverage, and measurement qualities such as SNR. M-STAF is effectively a tabular format to express all the measurement requirements for each science investigation in a way that enables cross-referencing across instruments and datasets, gap identification in the requirements set, and communication of measurement needs in context.

Implementation

STAF is implemented in Microsoft Excel. It is owned and maintained by the Project Science team, with the Project Scientist owning the P-STAF matrix and individual investigation scientists owning a given investigation's M-STAF matrix. The P-STAF is also released as a project document in JPL's institutional database. It was developed in 2016 and has been in continuous use since then.

Benefits

STAF has been shown to add value to the project in a number of ways by:

- Offering a framework for rigorously demonstrating the completeness and consistency of science and measurement requirements across the large number of investigations.
- Improving traceability of engineering requirements to science requirements, allowing a more efficient analysis of design sufficiency
- Enabling a unique and powerful set of analyses and fault studies to determine science return on different implementation options. This is traditionally a very difficult goal.
    - STAF played a major role in improving the efficiency of tour evaluation. Using a combination of P-STAF and M-STAF, the tour designers were able to rapidly evaluate each generated Europa Clipper tour, allowing more iterations and better screening for science return in the trajectory design.
    - Similarly, STAF formed the basis of probabilistic risk assessment to evaluate the impact of environmental threats such as radiation to the chances of mission success, and evaluate the benefit of risk mitigation design changes such as autonomous flyby recovery [11]
- Enabling key architectural trades, including descopes and L1 augmentation, in a complex payload with many complementary contributions. Because the STAF illuminated more of the nuances in the science story, it became easier to quantify the science impact of architectural changes, and highlighted opportunities to rewrite L1 requirements to better align with stakeholder expectations
- Providing a neutral framework that structured conversations among scientists about how mission success derives from the individual instrument investigations. On this multi-instrument mission it was critical to be able to define and agree on these relationships early in development, and STAF served as common tool for recording and explaining them.

Challenges

While STAF ended up being a very successful tool, it did face a set of challenges in its development and implementation:

- Scientists had concerns that the information in STAF (P-STAF in particular) would be used without sufficient project science/Principal Investigator participation to interpret and contextualize it, especially in architectural and resource decisions. Overcoming this challenge required building confidence that any conclusions drawn from data in the STAF was controlled and vetted by the science owners of the information, and that the systems engineers would demonstrate appropriate sensitivity in its use across the project.
- STAF was developed after the MBSE-based requirements development described in Section 6 had been in place for almost a year. The AF had driven many rounds of batch changes to the measurement requirements as the framework evolved, without any obvious science benefit to the measurement requirements. As M-STAF offered yet another approach to managing requirements, we faced initial reluctance from the scientists derived from this "process fatigue." We addressed this challenge by first focusing on the power of this framework to identify concrete, highly-relevant gaps in their requirements set that could threaten their science. This value was much easier to demonstrate and was key to the adoption of the framework.

Lessons

STAF has demonstrated that simple but well-conceived tools can enable important conversations between engineers and scientists. It has also shown the significant value and incredible leveraging power that a functioning architecture offers – not just to systems engineers, but to scientists and other parts of the project as well. The promises offered by centralized, model-based information management are realized in efficiency increases in trades, requirements development and validation, and interface management. Its implementation does offer some insight into factors that help address success:

- Put the novelty where it really matters and keep the other parts as standard as possible. For example, as the STAF taxonomy was new, we chose to implement our matrices in Excel, which improved adoption of the tool and broadened its user base. The fact that everyone could read, edit, and maintain the STAF matrices improved the quality of the database while the transparency of the tool built confidence in their reliability.
- Let key stakeholders across the project (not just

systems engineers) define the level of complexity needed. Rather than starting with a pre-made, intricate framework that could anticipate every possible need, we started with the simplest framework available and added complexity at the request of the stakeholders – increasing a sense of ownership and improving STAF's utility.

- Make sure the information and tools are owned by the right people. A significant contributing fact to the success of STAF is the fact that they are owned and maintained not by the engineers but by the scientists. As it is the Clipper's framework for articulating how the project science story is told, scientists must understand and believe in the STAF matrices so they use them, defend them, and maintain them. A tool made by systems engineers for systems engineers would have less reach and less support.
- Finally, allow the data (and its stakeholders) to guide the development of the database, rather than imposing a structure and trying force the data into it. For science systems engineering in particular, complexity of natural phenomena translates in the complexity of how different measurements relate to each other to test hypotheses. As part of the STAF effort, it was clear to the engineers that designing a framework that could capture some of these complexities provided unprecedented insight into how to meet the L1 Mission requirements.

8. OpenCAESAR Semantic Modeling Platform

Recognizing the risks of having every project develop its own unique system engineering practices, JPL's CAESAR project was founded in 2017 to establish some common modeling practices and tools that could be sustained across projects. Sponsored by JPL's Engineering and Science Directorate, the project developed out of over a decade of research and development and some practical experiences learned from the Europa Clipper modeling effort. To keep the development work focused, the project planned a roadmap that aimed to deliver some specific services needed by the Europa Clipper project while also developing some supporting frameworks and practices that would enable developing similar applications in the future. A subset of these tools was put into open source to facilitate external collaboration.

The CAESAR architecture takes a general metamodeling approach based on semantic web standards -- specifically the Web Ontology Language [12] with Description Logic (OWL2-DL). While this approach starts at a more general level of abstraction than, say, SysML does, it permits the development of simpler and more precise modeling vocabulary that can more concisely describe a system of interest (while SysML is intended for system engineering its vocabulary is very general and needs specialization to say almost anything precisely enough to be automatically validated). Unlike other modeling frameworks, OWL is fundamentally designed to enable precise, semantically rich, description, and automated transformations and reasoning. OWL is supported by many free and commercial tools including generic reasoners that can be used to validate the consistency of OWL models.

OpenCAESAR [13] extends OWL by defining an even more concise syntax called Ontological Modeling Language (OML) that encodes certain commonly-used patterns from OWL2-DL in order to help ensure that models are correct by construction, and to facilitate model management and continuous integration (CI) of OML models (for system engineers not familiar with the concept of CI the idea comes from the software world where best practice is to use automation to compile and test code as new modules are committed by multiple authors in order to find out about discrepancies as soon as possible.)

OpenCAESAR provides a set of example vocabulary that defines some common concepts and relations used in system engineering in the form of OML ontologies. OpenCAESAR provides tools to validate the consistency of these vocabularies and generate documentation from them (documentation is published on the OpenCAESAR website). These vocabularies can be extended and specialized for particular disciplines, or you can start from scratch and build your own following these examples, and using OpenCAESAR tools.

OpenCAESAR tools begin with the OML API, a java API based on the Eclipse Modeling Framework and XText that can be used to build tools that natively understand OML. The Rosetta OML Workbench is an Eclipse/EMF desktop application that provides an integration of OML editing and analysis features. Several other tools encode some common transformations and analysis as Gradle tasks that can be invoked in a continuous integration process. These include a task to convert OML to OWL, perform various analyses including merging models, inference using a generic reasoner, loading the resulting model data into a graph database, performing queries against that database to produce reduced datasets, and then rendering those datasets into HTML, or PDF document viewpoints, or into other export datasets in XML, CSV, JSON, or other syntax. Most interestingly, OpenCAESAR leverages Gradle's dependency resolution mechanism to publish and retrieve vocabularies and model fragments in the same way that it can publish and retrieve software tool updates. This deployment mechanism and the use of git and Github (OpenCAESAR is hosted in Github) for configuration management have been key enablers for massively distributed open source projects, but they also effectively scale down to single-developer projects, making them good candidates to use here.

Benefits

OpenCAESAR leverages open standards and widely-used tools and methods from the software development community to solve system engineering problems. This

enables easy integration of a wide variety of COTS and open source tools beyond the ones specifically used in the OpenCAESAR toolkit.

Challenges

Sustaining customized tooling and methods turns out to be quite challenging at JPL because of JPL's status as a Federally Funded Research and Development Corporation (FFRDC). JPL's organization funnels most funding directly to flight projects whose fundamental goal is development of an operational flight system. These projects have neither the resources nor the mandate to develop cross-project tools. Working with the limited development funding available for these tools is one of the key challenges facing the CAESAR team.

Starting an open source project is no trivial matter, but this proved to be the only way we could effectively collaborate with external partners due, once again, to FFRDC contracting limits that prevent us from having internal work paid for by other partners.

9. CAESAR Electrical Harness Specification Tooling

When JPL decided to establish the CAESAR project its initial plan included the development of a specific application to support the specification of electrical harnesses for the Europa Clipper spacecraft. This was a strategic decision intended to help keep the CAESAR effort focused on real project needs while at the same time attempting to mitigate what was known from past experience to be a highly laborious and error-prone process in time to help Europa.

Harness specification is a function performed at the spacecraft system level intending to specify requirements for cabling that will connect interfaces between assemblies of different subsystem (assuming that any connections between assemblies of the same subsystem can be specified and supplied by the subsystem, but inter-subsystem junctions require system scope). These specifications are delivered to a harness supplier who will elaborate a design, implement it, and deliver the harnesses to either the testbed or ATLO for integration.

The development effort started with a spreadsheet that had been repurposed from an earlier project and populated with Clipper data. The CAESAR team extended its existing vocabularies to capture the key things and relations inferred from the spreadsheet, and then developed some authoring viewpoints around them in the Eclipse/EMF authoring tool. These authoring viewpoints were designed to look (at least initially) very similar to the spreadsheet that users were used to in order to minimize the need to retrain users (subsequently, additional views were added to support other modes of editing). Figure 8 shows a screenshot of the functional assembly model organized by subsystems. This model is also used to assign unique (reference designator) identifiers to the assemblies, and is thus called the Reference Designator view. Figure 9 shows a screenshot of the electrical function list view. This view allows connections between assemblies selected from the composition model to be specified.

The specification process typically evolves from first identifying demands such as electrical loads, and sensors that need to be connected to an i/o device. These demands are initially mapped to a subsystem that will be responsible for satisfying them (e.g., the Power subsystem gets all the power load demands). Once the supplier subsystem identifies a target assembly the demands can be mapped to a specific assembly, and then to a particular "end circuit" interface such as a switch or I/O channel. The model explicitly supports this incremental specification and provides rules for incremental validation.

Authoring is only the first step in the workflow for this process. The initial delivery of the Electrical Flight System Engineering (EFSE) workbench saves its model as an Eclipse

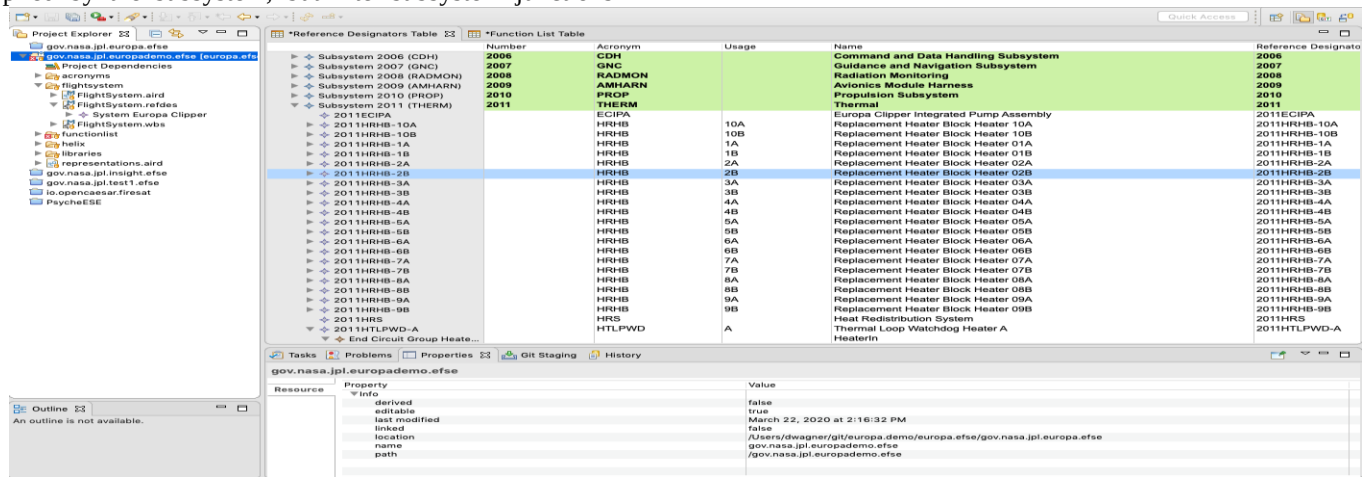


Figure 8. Functional composition view

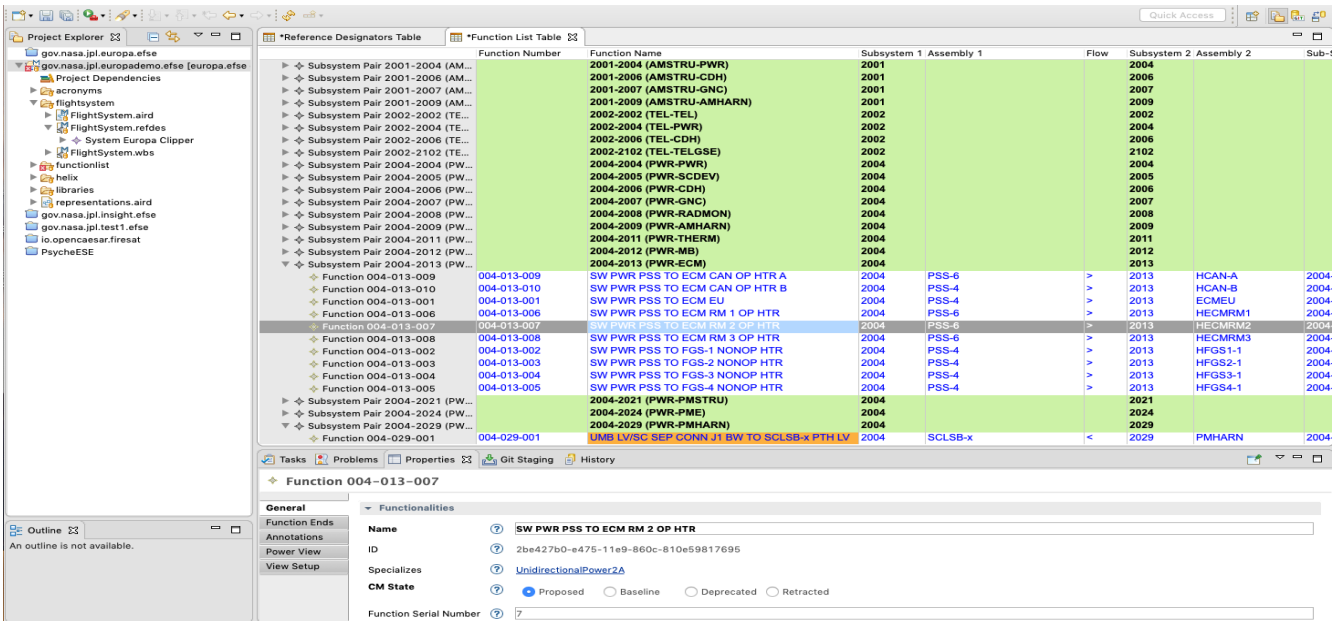


Figure 9. Function list view

EMF/XMI file which is committed to a Git repository for configuration management. This is facilitated by the fact that Eclipse has a native Git integration. The Europa EFSE team now use a standard Git CM process where model updates are performed on task-specific branches and merged into the baseline model by a team lead. Change tasks are controlled by the project's change control process.

Every time a model change is merged a continuous integration workflow is launched to convert the model to OWL, perform the analysis and inferencing needed to validate the model, load the data into a graph database, and then query that data into various analytical and transformation viewpoints to produce reports, documents, and derived data products. Some of those derived data products feed into the design model.

CAESAR provides a web console that indexes the reports generated by a workflow execution in order to facilitate users finding the report they need. Reports are generated such that distinct versions of every report are linked to the specific workflow that produced them so that the provenance of every report is explicitly reviewable, and any web link to a specific report will always return exactly the same report. See Figures 10 and 11.

More details about the CAESAR harness specification were previously reported in [14].

Benefits

Compared to the ad-hoc methods used in the past, the CAESAR tooling and workflow significantly improved the efficiency and effectiveness of the harness specification process by providing more immediate feedback to users about simple structural inconsistencies -- in many cases preventing those from being committed in the first place.

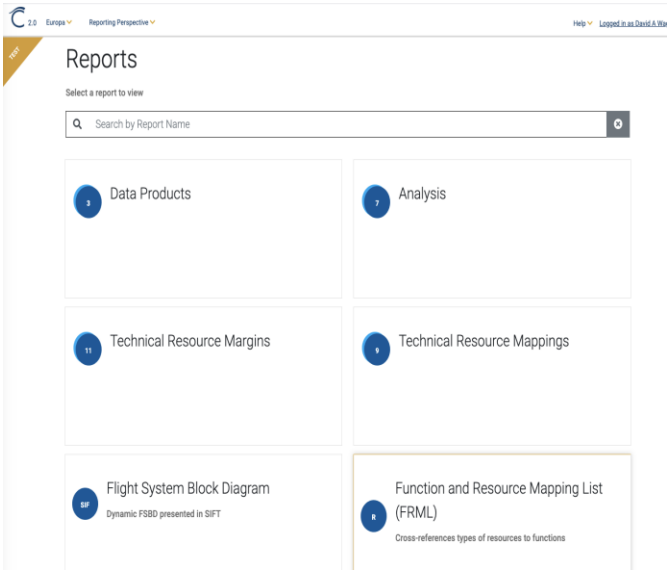
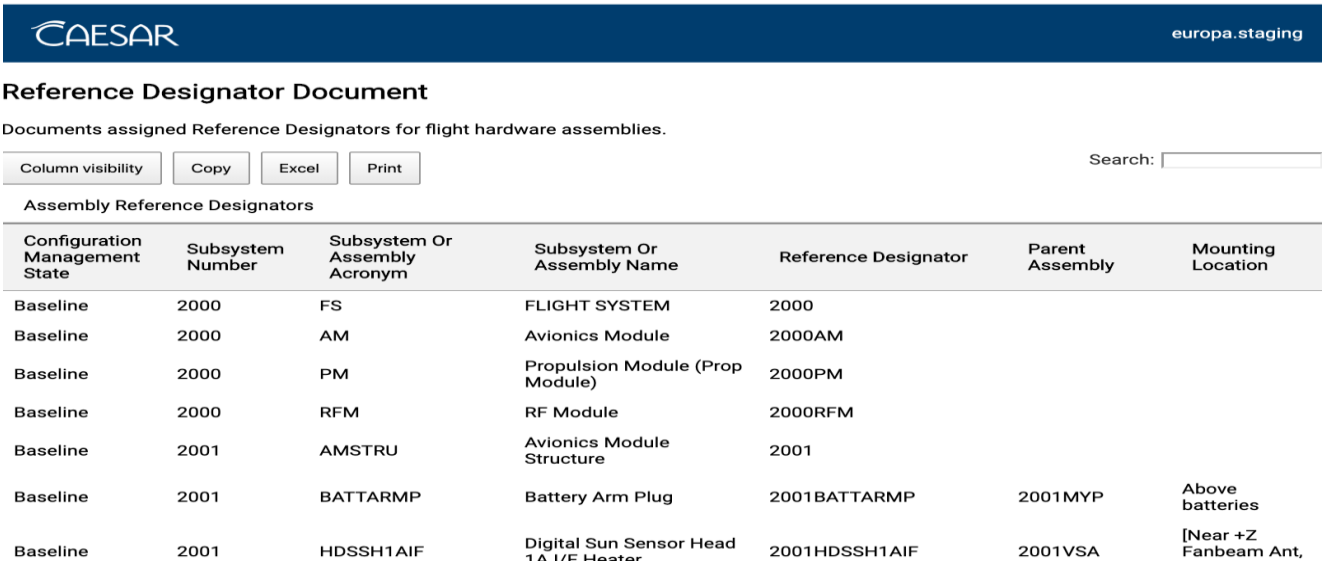


Figure 10. CAESAR Console reports index

Users found it particularly helpful that the authoring tool supported common refactoring operations such as renaming/removing/replacing assemblies while ensuring referential integrity.

A secondary benefit is that most of the toolchain, including the workbench, vocabularies, adapters, analysis, and reports, was developed to be reusable by other projects. Shortly after the system was deployed for Europa the Psyche project began using the system. As intended by the architecture, all of the project customization was limited to development of a project-unique model library that defined types of assemblies


CAESAR europa.staging

Reference Designator Document

Documents assigned Reference Designators for flight hardware assemblies.

Column visibility | Copy | Excel | Print | Search:

Assembly Reference Designators

Configuration Management State	Subsystem Number	Subsystem Or Assembly Acronym	Subsystem Or Assembly Name	Reference Designator	Parent Assembly	Mounting Location
Baseline	2000	FS	FLIGHT SYSTEM	2000		
Baseline	2000	AM	Avionics Module	2000AM		
Baseline	2000	PM	Propulsion Module (Prop Module)	2000PM		
Baseline	2000	RFM	RF Module	2000RFM		
Baseline	2001	AMSTRU	Avionics Module Structure	2001		
Baseline	2001	BATTARMP	Battery Arm Plug	2001BATTARMP	2001MYP	Above batteries
Baseline	2001	HDSSH1AIF	Digital Sun Sensor Head 1A I/F Heater	2001HDSSH1AIF	2001VSA	[Near +Z Fanbeam Ant,

Figure 11. Example HTML report produced by CAESAR workflow

and interfaces unique to the project, and a few custom reports that specifically filtered for those project-specific types. The logic for ⅔ of the reports was standardized so that it is now project-independent.

Challenges

Developing new tools and methods while they are actively being used on a project was a known challenge from Europa and earlier experiences, but was deemed necessary to help keep the product focused on the most immediate needs. This was probably most effective at the beginning of the effort because as the project work progressed it became more difficult to change the tooling or model architecture to accommodate newly-discovered disconnects without impacting project deliverables. Thus, the effort accumulated a large backlog of lessons learned or improvements that should be made before the next project customer starts using the tooling, but that cannot be pushed to the current customers.

CAESAR employs a strong Git-based model CM process that was, at first, a bit foreign to users who were nonetheless very familiar with the configuration management problems they were encountering with their spreadsheet process. CAESAR provided focused training and documentation as well as personal support to get users past this obstacle. Users quickly learned the procedures and best practices and this feature quickly became one of the most widely-praised aspects of the whole CAESAR experience.

A more subtle challenge emerged later in the process when the CAESAR team learned that a large amount of data the users were inputting in the tool was being manually transcribed from interface control documents (e.g., connector and pin mappings). A project may have millions of such bits of information and the effort involved in cutting and pasting this data is a huge amount of effort (not even counting the work it took to put it into all those documents in the first place), and more work is needed to continually validate consistency between the downstream design and the many ICDs as the process proceeds. Formal process still requires that some of the outputs of the EFSE process be published in documents, and so we have observed engineers cutting and pasting data from CAESAR reports back into spreadsheets so that they can be published as formal documents.

Lessons Learned

Although users see very little of the underlying model vocabularies, this application demonstrates that they are an effective way to concisely capture relevant information, and that they CAESAR semantic web tools can effectively simplify the code needed to implement downstream analysis and transformation. (these lessons may not have been visible to the customer project since they didn't pay for much of the work, and the savings won't be realized until the next project reuses the tools).

The CAESAR architecture, and specifically, the core vocabularies, provide a way to effectively separate the tools from the model in a way that permits projects to have some flexibility to specialize the model without having to change the tools that interpret them. E.g., the core vocabulary understands the concept of an Assembly as a functional hardware component. Projects can specialize their model library to define project-unique types of Assemblies, and the workbench and reports will understand those specializations without any need to customize the software.

This tooling has substantially streamlined the effort needed to author and validate harness specifications by allowing engineers to focus on the substance rather than the syntactic details of the specification, preventing many common mistakes, and highlighting and suggesting fixes to others. Even more importantly, though, the process of developing

this capability has identified numerous process improvements that were subsequently integrated into the capability such as clarifications to the vocabulary to help distinguish between details that belong to specification (requirements) and ones that belong to design, separation of those properties into separate parts of the model, and analysis to compare design with specification to aid design validation. Improvements such as this have avoided rework that used to occur as the design matured. Furthermore, this capability has enabled some standardization of methods that will make the capabilities reusable across projects while still providing the level of configurability needed to model different spacecraft architectures.

10. Future Work, Summary and Conclusion

Europa Clipper, in partnership with IMCE, has provided a rich opportunity for innovation and learning. As we knew at the beginning, the results so far serve to remind us that change is difficult and seldom straightforward, and that progress requires patience and steadfast commitment

We are incorporating the lessons into our approach for the next ten years, with a new set of projects as proving grounds

We continue to build for the future – laying a solid foundation and exploring new methods that benefit NASA's mission and the aerospace industry as a whole. System architecting and design synthesis are two areas which deserve more focus – being key to providing innovative & effective solutions. JPL is particularly focused on architecture-centric design approaches; completeness, stability and validity of system requirements; and thorough, systematic behavior analysis. Increased use of Human-System Interaction (HSI) techniques is helping build more utility and usability into new tools and processes. We are formally adopting a strategy of incrementally building capabilities over multiple project lifecycles - incremental improvements on one project that can be leveraged and grown on the next. We are developing an architecture to enable information exchange by engineering teams – based on analysis of user needs & wants. We continue to invest in a mix of custom and COTS tooling to support these processes and methods, endeavoring to use COTS where possible, to leverage the investments of others.

We will continue to push the SE envelope and learn our lessons along the way

Acknowledgements

The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004).

References

[1] Model Based Systems Engineering On The Europa Mission Concept Study, T. J. Bayer, S. Chung, B. Cole, B. Cooke, F. Dekens, C. Delp, I. Gontijo, K. Lewis, M. Moshir, R. Rasmussen, D. Wagner, IEEE Aerospace Conference Proceedings, 2012

[2] Model-Based Systems Engineering Approach to Managing Mass Margin, S. Chung, T. Bayer, B. Cole, B. Cooke, F. Dekens, C. Delp, D. Lam, Proceedings of the 5th International Workshop on Systems & Concurrent Engineering for Space Applications (SECESA), 2012

[3] Early Formulation Model-centric Engineering On NASA's Europa Mission Concept Study, T. J. Bayer, S. Chung, B. Cole, B. Cook, F. Dekens, C. Delp, I. Gontijo, K. Lewis, M. Moshir, R. Rasmussen, and D. Wagner, Proceedings of 22nd Annual International Symposium (IS2012), 2012

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

[5] Cloud-based orchestration of a model-based power and data analysis toolchain, E. Post, K. Dinkel, E. Lee, B. Cole, H. Kim, B. Nairouz, IEEE Aerospace Conference Proceedings, 2016

[6] A Case for Model-Based Architecting in NASA, Rasmussen, R. and Muirhead, B., August 2012, CL#12-4124, NASA/Caltech

[7] Architecture Modeling on the Europa Project, G. Dubos , S. Schreiner , D. Wagner , G. Jones , A. Kerzhner , J. Kaderka, AIAA Space Conference Proceedings, 2016

[8] A Framework for Writing Measurement Requirements and its Application to the Planned Europa Mission, S. Susca, L. Jones-Wilson, B. Oaida, IEEE Aerospace Conference Proceedings, 2017

[9] A Framework for Extending the Science Traceability Matrix: Application to the Planned Europa Mission, L. Jones-Wilson, S. Susca, IEEE Aerospace Conference Proceedings, 2017

[10] Project-domain Science Traceability and Alignment Framework (P-STAF): Analysis of a Payload Architecture, L. Jones-Wilson, S. Susca, IEEE Aerospace Conference Proceedings, 2018.

[11] Assessing Science Robustness of the Europa Clipper Mission: Science Sensitivity Model, L. Jones-Wilson, S. Susca, B. Bradley, IEEE Aerospace Conference Proceedings, 2018.

[12] Web Ontology Language https://www.w3.org/OWL/

[13] OpenCAESAR Website https://opencaesar.github.io

[14] Wagner, David, et al. "CAESAR Model-Based Approach to Harness Design." *2020 IEEE Aerospace Conference*. IEEE, 2020.

BIOGRAPHY

Todd Bayer *is a Principal Engineer in JPL's Systems Division. He is currently the Chief Engineer for the Flight Systems Engineering, Integration and Test Section at JPL. He received his B.S. in Physics in 1984 from the Massachusetts Institute of Technology. He started his career as a project officer in the US Air Force at Space Division in El Segundo, California. Following his military service, he joined the staff of JPL in 1989. He has participated in the development and operations of several missions including Mars Observer, Cassini, Deep Space 1, and Mars Reconnaissance Orbiter, for which he was the Flight System Engineer for development and Chief Engineer during flight operations. During a leave of absence from JPL, he worked as a systems engineer on the European next generation weather satellite at EUMETSAT in Darmstadt, Germany. From 2010-2020 he served as the Flight System Engineer for Europa Clipper, and before that was Assistant Manager for Flight Projects of JPL's Systems and Software Division.*

John Day *has over 25 years of systems engineering experience working on a variety of NASA projects. He has provided key technical and managerial expertise to human, deep-space, and robotic observatory projects and programs (Europa Clipper, Constellation, Mars Science Lander, Mars Reconnaissance Orbiter, Deep Impact, Kepler, SIM, Spitzer, Cassini and Galileo). He is actively involved in advancing the theory and state of practice of systems engineering. John's experience includes working at his own systems engineering consulting company, at Lockheed Martin Missiles and Space, and at JPL. He is currently the Manager of the Engineering Development Office in the JPL Systems Engineering Division, and the Lead for the JPL Center for Autonomy.*

Emma Dodd is *a System Engineer in the Project System Engineering and Formulation Section at the NASA Jet Propulsion Laboratory in Pasadena, CA. She is currently the Technical Resources Lead on the Europa Clipper Flight System Team. She is responsible for the instruments and subsystems power and mass update process, threats and opportunities, and margin reporting along with the peak power and current analysis. She received her bachelor's degree in Mechanical Engineering from Brown University and a master's degree in Aerospace Engineering from the University of Southern California.*

Laura Jones-Wilson *earned her Ph.D. from Cornell University in Aerospace Engineering specializing in Dynamics and Controls in 2012. She then joined JPL as a guidance and control systems engineer, where she served in roles including the PI on a Mars Sample Capture technology development effort and the co-manager of the SmallSat Dynamics Testbed. She joined Europa Clipper as the instrument engineer for UVS and later became the IE for REASON. After serving in the PSE office as the lead systems engineer for the REASON-Solar Array Integrated Product team, she is now the Payload V&V Lead for Clipper.*

Andres Rivera *is a Systems Engineering in the System Modeling and Methodology group at the NASA Jet Propulsion Laboratory, where he has worked as part of the Europa Clipper Project System Engineering Team since 2017. He is responsible for the end-to-end modeling infrastructure supporting technical resource estimation, design capture and other systems modeling efforts. He holds a master's degree in Mechanical Engineering from California State University Long Beach, his thesis focused on the modeling and robustness analysis of formations of time-delayed agents. He received his bachelor's degree in Mechanical Engineering from The City College of New York in 2016.*

Dr. Narek Rouben Shougarian *is a Software Systems Engineer at the NASA Jet Propulsion Laboratory where he has worked on Europa Clipper from 2017. His responsibilities include system power modeling, leading and supporting trade studies as well as supporting and leading probabilistic risk assessments. He received his doctorate in Aerospace Systems from the Massachusetts Institute of Technology in 2017. His PhD thesis focused on the automatic generation, optimization and evaluation of different hybrid electric aircraft propulsion systems and included a case study on electronic devices. He completed his MEng thesis at Imperial College London in 2011. Dr. Shougarian has experience in system architecture at JPL as well as from work with a number of industry and government research partners at MIT, including Google, DARPA, Pratt & Whitney and the US NAVY.*

Sara Susca *received her PhD in Electrical Engineering from UCSB in 2007. She spent three years at Honeywell Aerospace developing new technology for GPS denied navigation. She has been with JPL since 2011 where she covered various roles including project manager for STABLE, instrument engineer for the EIS instruments and PIMS instrument for Europa Clipper. Currently she is the Payload Systems Engineer for SPHERx.*

David A. Wagner *manages both the System Modeling and Methodology group as well as product development on the CAESAR project at the Jet Propulsion Laboratory. He holds a bachelor's degree in Aerospace Engineering from the University of Cincinnati, a master's degree in Aerospace from the University of Southern California, and over forty years of experience in systems engineering large and small projects at JPL.*