Architecture to Geometry – Integrating System Models with **Mechanical Design**

Dr. Manas Bajaj¹ and Dr. Dirk Zwemer² Intercax, 75 5th Street NW, Suite 312, Atlanta, GA 30308, USA

Dr. Bjorn Cole³ NASA Jet Propulsion Laboratory (JPL), California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA 91105, USA

Model-Based Systems Engineering is founded on the principle of a unified system model that can coordinate architecture, mechanical, electrical, software, verification, and other discipline-specific models across the system lifecycle. This vision of a Total System Model as the digital blueprint (or digital twin) of a system, federating models in multiple vendor tools and configuration-controlled repositories, has gained tremendous support from the practitioners. A software platform, Syndeia, developed by Intercax, provides capabilities for seamless model-based communication between systems engineering and X (where X =mechanical/electrical, simulation, PLM, ALM, project management, and other disciplines), replacing the existing document-centric approaches. This paper elaborates research and development performed by NASA JPL and Intercax for integrating system architecture models (SysML) and mechanical design models (CAD) with applications to the Europa Clipper Mission. Specifically, this paper demonstrates (1) seeding of mechanical design models from system specifications (SysML) as a starting point for mechanical design, (2) model-based connections between system and mechanical design parameters, including compare and bi-directional synchronization, (3) abstracting system architecture from mechanical assemblies for transitioning existing/old projects to a model-based systems approach, and (4) use of persistent, fine-grained connections between system architecture and mechanical design models for continuous verification and communication between the two disciplines. The paper also covers organizational, cultural, and technical challenges that need to be addressed for seamless integration between system architecture models and mechanical/electrical design models, as well as other disciplines.

Nomenclature

ALM	=	Application Lifecycle Management
CAD	=	Computer-Aided Design
CAE	=	Computer-Aided Engineering
DBSE	=	Document-Based Systems Engineering
MBE	=	Model-Based Engineering
MBSE	=	Model-Based Systems Engineering
PLM	=	Product Lifecycle Management
SE	=	Systems Engineering
SysML	=	Systems Modeling Language

¹ Chief Systems Officer, Intercax, AIAA Non-Member

² Chief Executive Officer & President, Intercax, AIAA Non-Member

³ Lead, Europa Project Systems Engineering Analysis team, NASA Jet Propulsion Laboratory, AIAA Member

I. Model-Based Systems Engineering

Mechanical and electrical engineering have both enjoyed the benefits of advancing computer technology. Advancements in computer-aided design (CAD) tools and product lifecycle management (PLM) environments make it possible to conceive and develop complex products in shorter time frames. Advanced computer-aided engineering (CAE) tools, such as math solvers and physics-based simulation tools, support rapid prediction of performance and design space exploration. Systems engineering, by contrast, has mostly been left out.

The primary tasks of a system engineer are:

- 1. developing constraints (requirements development, architecture, and verification)
- 2. segmentation (architecture, interfaces, functional decomposition)
- 3. removal and recombinations of detail (views of the system, scenarios)
- 4. appreciation of interaction (integration, test, validation, feedback, and behavior)

These so far have been practiced without much tooling, typically facilitated by exchanging engineering memos, mechanical drawings and sketches, and spreadsheets. Model-based systems engineering (MBSE) aims to improve this by providing computer tools that enhance all of these practices.



Figure 1. Goal of MBSE is to create a unified system model from which SE artifacts can be auto-generated

The goal of MBSE^{1,2} is to create and manage a single unified model of the system that can represent all of its varied aspects, such as requirements, structure, and behavior, as shown in Figure 1. The system model is conceptualized as a graph of information that can be viewed from or projected onto different perspectives. The transition from document-based systems engineering (DBSE) to MBSE is similar to the transition from 2D mechanical drawings to 3D CAD and the ability to automatically derive 2D views from a 3D CAD model.

The Systems Modeling Language (OMG SysML)³ has emerged as an open, international, and industry standard for representing the system architecture. However, most of the detailed engineering is performed in a variety of modeling and simulation tools, as shown in Figure 2. These include, but are not limited to, mechanical / electrical CAD tools; CAE tools (FEA - finite element analysis and CFD - computational fluid dynamics); mathematical modeling and simulation tools; and a variety of enterprise

repositories and databases, such as product lifecycle management (PLM) for hardware design and manufacturing, application lifecycle management (ALM) for software design/build and verification, requirements management, and project management.

Figure 2 shows a federated approach to realize the vision of MBSE. The SysML model is used to represent the high-level architecture of the system. Elements in the SysML model can then be connected via model-based connections, as shown by blue radial arrows, to elements in other engineering tools, such as CAD, CAE, and requirements management. These connections provide services for model transformation, comparison, and synchronization as all the models evolve in a concurrent development environment. Model transformations also provide an efficient way to automatically generate models and create inter-model connections, such as seeding a PLM part structure from a SysML block structure (or vice versa), or generating a simulation model from a SysML activity model (or vice versa). Hence, connections can be created between existing models or generated during model transformations.



Figure 2. Combining system architecture with disciplines to realize the MBSE vision



Figure 3: Total System Model is the digital blueprint of a system 3

Enterprises developing complex systems use a variety of version control and configuration management systems, such as PLM systems for versioning and managing the engineering/manufacturing bills-of-materials, and ALM systems for versioning and managing software code, builds, and related artifacts. The federated system model includes models managed in different version control systems and inter-model connections between elements of these models. This approach provides organizations the flexibility to use the best-in-class tools for each discipline and connect them to other models in the federation. The conception, development, and management of a federated system model is a fundamental shift underway in organizations developing complex systems.

In our work, we refer to the federated system model as **The Total System Model (TSM)**, as shown in Figure 3. The TSM is a digital blueprint of the system that evolves through the lifecycle as the participating models evolve (Figure 4). The TSM is aware of the versions of connected models in various configuration management systems (e.g. PLM, ALM, Requirements Management). Technical data packages and other system engineering artifacts can be automatically generated from the TSM at various milestones in its development timeline.



Figure 4: Total System Model evolves over time as individual models in the federation evolve

Syndeia⁴ is a software platform developed by Intercax that provides services to create, manage, maintain, visualize, and query the Total System Model. We will elaborate these capabilities in section IV.

This paper is organized in the following manner. In the next section, we discuss the current challenges with model-based practices in mechanical design and systems engineering, followed by an elaboration of use cases for integrating systems architecture and mechanical design models in section III. In section IV, we present Syndeia as a software platform for MBSE, and demonstrate how it addresses the system architecture – mechanical design integration use cases. In section V, we present the application of this research, including the Syndeia platform, for the planned Europa Clipper Mission being developed at NASA JPL. In section VI, we present a summary of this effort and in section VII, we present the lessons learned and potential areas of future work for this initiative.

II. Current Challenges

The mechanical CAD practice, despite using models within the team, is still very document-centric. The record of authority is the drawing, and the way to navigate this information is still a kind of table of contents, the document tree. The drawings have a preview, review, and release process that takes up comments and corrections before the authoritative release. After this release, revisions are provided by formal change requests.

The drawing-based release cycle shows how using models does not translate into model-based practice, at least not outside the mechanical team. The separation of the view from the model allows the model to be in a wide variety of states during its development. Even on large projects at JPL mechanical design teams tend to be relatively small (less than 10 people), which means the state of the model can be well-understood by its developers. There are no guarantees that the model is converged or even usable at any time other than when a new drawing or other product is generated. The working state of the model may have one change from a baseline or many, and it is unknown whether or not more changes are intended.

Systems engineering is also in transition from document-based to model-based practice. With the adoption of SysML, organizations are increasingly developing system architecture models that can be queried and analyzed versus documenting the system during/after development. However, the larger challenge to integrate the system architecture model with various disciplines remains. In this paper, we focus on approaches and tools to address this major challenge.

Model-based practice for systems engineering is gradually taking inspiration from the source code version control practices of software. They key to these practices is that someone can look at the repository and be confident that they are accessing a working, relevant version of the model that needs a new feature or improvement. Stable, accessible baselines are established early and often as groups of changes are incorporated. Technology efforts are being put forward to enable the branch-and-merge style of incorporating changes that are common in tools like Git⁵. These techniques greatly increase the confidence that a knowledgeable user can understand the state of the model without being directly embedded in the core team. Indeed, Git is known for its strengths in supporting geographically distributed teams.

For system-wide analysis support, a parameterized CAD model is much more useful than an un-parameterized one. Unfortunately, parameterization is much more labor intensive in the setup and can be brittle in feature-based CAD systems because parameters often are directly attached to features. There are techniques to make a parameterized model more robust, but again these require greater effort and planning to achieve.

In order to put mechanical CAD practice on model-based terms, not only is cultural change needed but additional effort needs to be put in. While the ability to electronically access the CAD model to feed other models reduces effort in the project as a whole, it will cause some increase to the efforts needed by the mechanical design team. This can make the design team look less efficient, and causes them to accrue the costs and risks without directly realizing the benefits. There are some altruistic crusaders for organizational efficiency, but it is a hard role to rely on.

III. Use Cases for Integrating System Architecture and Mechanical Design

In this section, we present five (5) main use cases for integrating system architecture models and mechanical design models.

A. Use Case 1 - Viewing and accessing CAD models from system architecture

The first use case for connecting system architecture models, such as in SysML, to 3D CAD models is to enable system engineers to access and open the CAD models to visualize the geometry, as shown in Figure 5 below. Specifically, if the SysML block representing the system-of-interest (or sub-systems) is connected to a CAD model, or the assembly or specific part in the assembly, managed locally or version-controlled in a PLM system, then system engineers can access and open the CAD model from the system architecture model in SysML. The ability to seamlessly access, visualize, and play (pan, zoom, rotate, measure) with the 3D mechanical design directly from the architecture model provides a much richer model-based experience compared to looking up spreadsheets for data or exchanging emails to gather information.



Figure 5. Access, open, and visualize 3D CAD models seamlessly from the system architecture

B. Use Case 2 - Seeding mechanical design models from system requirements (Requirements to Geometry)

System engineers formulating the specifications for hardware sub-systems in the overall system architecture need a model-based approach to transfer the specifications to mechanical and electrical design teams. In the traditional approach, system engineers push text-based requirements to a requirements management system and assign conformance to mechanical/electrical design engineers. The interpretation of text-based requirements by the design engineers is a potential source of error in communicating hardware requirements, leading to costly redesign.

A higher-fidelity model-based approach is presented here. If system engineers can represent hardware requirements related to geometry in a parametrized model-based way, then those requirements can be used to seed CAD models with geometric features representing the hardware specifications. This includes, for example, generation of features related to bounding box constraints, keep-out zones, mass/volume constraints, and part mating constraints. Instead of interpreting and manually verifying text-based requirements, mechanical/electrical design teams use the seeded CAD model as their starting point with all system-level constraints encoded as geometric or parametric boundaries. Not only can they visualize *system requirements as geometry*, the geometric engine in CAD tools can dynamically compute when they violate those constraints. Early detection of these violations can significantly reduce costly redesign and improve efficiency. Changes to existing, or addition of new, system-level requirements can be synchronized to the mechanical design model in a similar manner.

As shown in Figure 6, a SysML model representing the cylindrical envelope for the payload instruments (A) can be used to generate the cylindrical part feature in the CAD model (B). As mechanical designers partition and allocate this space for different instruments, they create new features boundaries (C) which can then be communicated back to the system architecture model, if needed (D). The ability to communicate hardware requirements as geometry to mechanical/electrical design teams is a fundamental enabler for the model-based vision.

One of the key capabilities needed for realizing this approach is the parameterized representation of system requirements⁶ in SysML, beyond the current text-based requirements. The SysML standards committee is currently working on property-based⁷ representation of requirements to be included in the next version of the SysML standard (1.5).



Figure 6. Seed mechanical CAD models from system specification

C. Use Case 3 - Access mass properties and geometric parameters for verifying system requirements

System engineers need to continuously track mass, size, center-of-gravity, packing factor, and other geometryrelated system attributes. Values of these geometric attributes are needed to support engineering analyses and automated verification of system requirements, such as mass roll-up calculations for verifying system mass requirements. During conceptual design these attributes need to be estimated from proof-of-concept 3D CAD models or conceptual analyses, while during detailed design these attributes need to be directly read from the latest version (or baseline) CAD model under version control.

Today, most CAD modeling environments can automatically compute mass properties of geometric parts and assemblies, such as mass, moments-of-inertia, volume, material volume, enclosed volume, bounding box size, and others. A model-based approach is required to facilitate the following:

 Connect attributes in the system architecture model to the mass properties in a CAD model such that changes in the geometry with new versions of CAD models can be tracked and synchronized to the system architecture model. This is illustrated in Figure 7 where the weight and bounding box system attributes are derived from and synchronized with values from the 3D CAD models. It is crucial to track units and facilitate automated unit conversions if the architecture and CAD models are based on different unit systems.



Figure 7. Parametrized connections are needed to track and synchronize system architecture attributes and mechanical part/assembly attributes

- 2. In addition to mass properties, system engineers and mechanical designers should also be able to communicate about key parameters in the 3D CAD model, such as dimensions of parts performing critical system functions, spatial constraints related to positioning of parts, or identifying keepout zones. These parameters may be defined at the assembly, part, or feature level in the CAD model. There are corresponding attributes in the system architecture model that need to be compared and synchronized with these key parameters in the CAD models.
- 3. From a lifecycle perspective, system engineers need to track the various states of the key parameters/attributes of interest. Consider the mass attribute of the system as an example. At the very least, system engineers need to track the system mass in the following states.
 - a. Required Mass (Max): This is the maximum allowable value of the system mass in the overall system requirement. For example, the mass of the system shall be less than 100 kgs.
 - b. *Budgeted Mass*: This is the maximum mass budgeted for the system. It is less than the required mass by a certain margin reflecting uncertainties and contingencies. For example, this value may be 95 kgs.
 - c. *Designed Mass*: This is the mass of the system, as computed from the latest version of the CAD model of the system. Mechanical designers have to ensure that this mass is within the budgeted mass by a certain margin to account for uncertainties and contingencies during manufacturing. For example, this value may be 94 kgs.
 - d. *Manufactured Mass*: This is the mass of the system that has been manufactured. This value must be less than the budgeted mass.

A similar approach to capturing and tracking the lifecycle states applies to other key parameters/attributes of interest.

D. Use Case 4 - Composing system architecture from CAD assemblies

For many organizations transitioning to model-based systems engineering, the best model-based definitions of various sub-systems are in their domain-specific models, such as CAD models, simulation models, and sometimes even spreadsheets. System engineers in these organizations are starting to develop a model-based definition of the

system architecture, such as by developing a SysML model of the system. For the hardware sub-systems, it would be convenient to reverse engineer the hardware architecture model from the CAD model and keep them connected, as shown in Figure 8 below.



Figure 8. Generate system architecture model elements from existing CAD models

The hardware sub-systems in the system architecture model can be related to software sub-systems to develop a unified architecture of the overall system. This system architecture model can now be the central model for orchestrating all the other disciplines together, beyond mechanical design.

E. Use Case 5 - Tracking system variants, states, and versions when coordinating with mechanical design

System engineers formulating system architectures also model system variants and states. For example, an unmanned aerial vehicle (UAV) system may have variants related to propulsion systems and fuselage/wing types. A solar panel on a satellite may be in the open-panel or closed-panel state. In contrast, the typical approach for managing variants and operational states for parts/assemblies in mechanical design are by creating separate CAD models. System engineers and mechanical designers need to be able to connect specific variants and states of the hardware sub-systems in the architecture model to the corresponding CAD models. A third dimension is versioning. Both the system architecture model and the mechanical design model (CAD) are simultaneously evolving during the development process and managed in different version-controlled repositories-architecture models in a dedicated database and CAD models in a PLM system. When connecting the system architecture model with the CAD model, it is important to track the versions of the models (or model elements) that are being connected.

IV. Syndeia for Model-Based Systems Engineering

Syndeia⁴ is a software platform for integrated model-based engineering (MBE). It enables engineering teams to collaboratively develop and manage the Total System Model (TSM) of a complex system (or project) by combining the system architecture model (SysML) with models in a variety of enterprise repositories and tools, such as PLM (e.g. Teamcenter⁸, Windchill⁹), CAD (e.g. NX¹⁰, Creo¹¹), ALM (e.g. GitHub¹²), Project Management (e.g. JIRA¹³), Requirements Management (e.g. DOORS-NG¹⁴), Simulations (e.g. Mathematica¹⁵ and MATLAB/Simulink¹⁶), Databases (e.g. MySQL¹⁷), and other data sources (e.g. Excel¹⁸).

Syndeia provides model transformation services to build the Total System Model graph, such as by dragging and dropping requirements from DOORS-NG to SysML, generating Simulink models and PLM part structures from SysML models, connecting behavior elements in SysML to software code in GitHub, or tracking the development status of a sub-system in JIRA directly from SysML. Syndeia provides services to search, visualize, compare, and bi-directionally synchronize connected models in the Total System Model graph, such as comparing and

synchronizing changes in system architecture to PLM part structure, or comparing connected SysML and Simulink models, or synchronizing changes in DOORS-NG requirements to SysML.

Syndeia was formerly known as SLIM^{19,20} (System Lifecycle Management) and was born out of a NASA SBIR program. Syndeia version 3.0 was released in July 2016. Although Syndeia started as a software tool for MBSE, it is developing into a software platform for model-based engineering (MBE)²¹ in general that can be used by all the disciplines and domain experts who need to interact with the Total System Model. In this section, we will demonstrate Syndeia's capabilities in addressing the five use cases for integrating system architecture and mechanical design models that were presented in section III.

The Syndeia Dashboard is the primary means to interact with Syndeia, and it can be launched directly from the SysML modeling environment, such as MagicDraw or Rhapsody. The Repository Manager tab, as shown in Figure 9, is used to connect to a variety of enterprise repositories—Teamcenter, Windchill, DOORS-NG, MySQL, JIRA, GitHub, or local/shared file systems—and models managed in these repositories—NX, Creo, Simulink, Excel, PLM part structures, requirement structures, projects, and issues. In Figure 9, the multi-level part-assembly structure in the NX CAD model of a toycar assembly can be seen. We will use the simple toycar assembly example to illustrate the concepts and then show more complex examples.



Figure 9. Syndeia Dashboard - Browse multi-level CAD assembly models

Figure 10 shows the Connection Manager tab in the Syndeia Dashboard with the SysML model on the LHS and the repositories on the RHS. A simple drag-n-drop operation from any element in the repository to a SysML element, or vice versa, can be used to generate models and create connections. The first type of connection is a Reference Connection. When selected during the drag-n-drop operation, as shown in Figure 10, it creates a reference connection between the source and target elements. A Reference Connection can be used to track changes in versions of the source and target elements.



Figure 10. Syndeia Dashboard - Drag and drop CAD parts/assemblies to the SysML model to generate models and/or create connections. The behavior depends upon the type of connection.

The second type of connection relevant for system architecture and mechanical design integration is a Model Transform connection. Selecting this type of connection and then dragging a CAD assembly model to a SysML package generates a SysML block structure corresponding to the multi-level CAD assembly structure. Model-based connections are created between the SysML blocks and the individual parts in the assembly, as shown in Figure 11.

🚭 Syndeia Dashboard (3.0.0) - NX Test				- 🗆 ×
🚆 Repository Manager 🛛 🗟 Connecti	ion Manager 🛛 🕌 Connection Browser 🛛 😹	Connection Summary 🛛 👫 Comparison Result 📉	Settings	
Q- Type here to filter connections				Export to Excel
Source (SysML Element) \sim	√ Name ∨	Type 🗸	Target \checkmark	Target Repository 🗸 🖽
⊡- 🛅 System				^
	[System] - [toycar_assy]	REFERENCE_CONNECTION	toycar_assy	NX CAD
toycar_assy				
	[System::toycar_assy] - [toycar_assy]	BLOCK_NX_PART_MODEL_TRANSFORM_CONNECT	toycar_assy	NX CAD
toycar_axle				
<u>_</u>	[System::toycar_axle] - [toycar_axle]	BLOCK_NX_PART_MODEL_TRANSFORM_CONNECT	toycar_axle	NX CAD
toycar_axle_assy				
	[System::toycar_axle_assy] - [toycar_axle	BLOCK_NX_PART_MODEL_TRANSFORM_CONNECT	toycar_axle_assy	NX CAD
toycar_body				
	[System::toycar_body] - [toycar_body]	BLOCK_NX_PART_MODEL_TRANSFORM_CONNECT	toycar_body	NX CAD
ing toycar_wheel				
	[System::toycar_wheel] - [toycar_wheel]	BLOCK_NX_PART_MODEL_TRANSFORM_CONNECT	toycar_wheel	NX CAD
				U
				*
[17:34:22] INEO Connection of type "BLOCK N	X PART MODEL TRANSFORM CONNECTION	and id "f1968980-ea55-44a5-a833-19ea7d18ab79" is	created (or loaded) betwee	n artifact(s) [SysML model
element "R2::toycar_axle_assy"] and [NX part	"toycar_axle_assy"], and added to connection	n manager.	e conce (c. loudea) bernee	
[17:34:22] INFO Connection of type "BLOCK_N	X_PART_MODEL_TRANSFORM_CONNECTION	" and id "1d091822-c4cf-4dbc-8c1f-73440c303b9a" is (reated (or loaded) between	artifact(s) [SysML model
element K1::Cylinder J and [NX part Cylinder [17:34:22] INEO Connection of type "BLOCK_N	J, and added to connection manager.	"and id "c5170533-176d-45ed-9e81-f6ff9c524622" is	created (or loaded) between	artifact(s) [SysML model
Ready			9:3	9:38 PM 312M of 66 1M 👘

Figure 11. Syndeia Dashboard - Connections automatically created between SysML blocks and CAD assembly parts when the SysML block structure was generated from CAD assembly structure

The resulting SysML block structure is shown in Figure 12 and Figure 13. For each assembly (or sub-assembly), the CAD to SysML model transformation process generates block part properties corresponding to assembly components, and value properties corresponding to CAD mass properties and user-defined parameters.

«block» «NX_Part» toycar_assy
SPARE_WHEEL : toycar_wheel parts REAR_AXLE : toycar_axle_assy BODY : toycar_body FRONT_AXLE : toycar_axle_assy FRONT_AXLE : toycar_axle_assy
values values values «NX_Mass_Property» surface_area : Real = 6323.540319416099{base_Property, unit = "mm^2"} «NX_Mass_Property» mass : Real = 0.11741058601840965{base_Property, unit = "kg"} «NX_Mass_Property» volume : Real = 14993.740743848475{base_Property, unit = "kg/mm^3"} «NX_Mass_Property» density : Real = 7.83064000000002E-6{base_Property, unit = "kg/mm^3"} «NX_Mass_Property» center_of_gravity : String = -2.594464966530994,-14.885710840868393,16.38294060133932{base_Property, unit = "mm"} «NX_Mass_Property» lower_left_z : Real = -0.3759351939541311{base_Property, unit = "mm"} «NX_Mass_Property» lower_left_x : Real = -0.3759351939541311{base_Property, unit = "mm"} «NX_Mass_Property» lower_left_x : Real = -0.375972873856{base_Property, unit = "mm"} «NX_Mass_Property» lower_left_y : Real = -30.32075272873856{base_Property, unit = "mm"} «NX_Mass_Property» lower_left_y : Real = -2.886888294579413{base_Property, unit = "mm"} «NX_Mass_Property» lower_left_y : Real = -2.7077820281117567{base_Property, unit = "mm"} «NX_Mass_Property» upper_right_z : Real = 27.131199977242446{base_Property, unit = "mm"}

Figure 12. Part/assembly mass properties and parameters are generated as value properties of the SysML block corresponding to the part/assembly

For the toycar model shown in Figure 9, part properties corresponding to the front axle assembly, rear axle assembly, body, and the spare wheel were generated for the components in the toycar assembly that was dragged and dropped on a SysML package. Mass properties such as surface area, mass, volume, density, center of gravity, and the coordinates of the extreme ends of the bounding box were also generated as value properties with default values, as shown in Figure 12. For each part property corresponding to an assembly component, an association block was created in SysML to represent properties specific to that component, such as its placement in the assembly. As shown in Figure 13, the toycar_assy block represents the toycar assembly and the toycar_axle_assy block represents the axle assembly used twice (front and rear) in the toycar. The two usages are represented as two part properties. The FRONT_AXLE part property is shown in an expanded form and the REAR_AXLE part property is shown in the parts compartment of the toycar_assy block. The association block FRONT_AXLE is anchored to the FRONT_AXLE part property and has value properties corresponding to the translation and rotation matrix representing the placement of the axle assembly as the front axle in the toycar.



Figure 13. SysML part properties and association blocks are generated for CAD assembly components

Figure 14 shows a more complex Creo CAD assembly of a RC Piper Cub J3 airplane²² and the generation of SysML block structure for the same using Syndeia. This assembly has over 450 individual parts/sub-assemblies.



Figure 14. SysML block structure generated for RC Piper Cub J3 plane with 450+ parts/sub-assemblies

Figure 15 and Figure 16 illustrate the opposite process. Starting with a SysML model with specification of primitive geometries, Syndeia can be used to generate a CAD model with features. For example, the My System block in Figure 15 uses several blocks representing primitive geometric shapes, stereotyped as NX part features, as part properties. When the My System block is dragged and dropped onto a folder in the local file system, as shown in Figure 16, an NX CAD model is generated with the specified features. This scenario addresses use case 2 presented in section III.B.



Figure 15. SysML model representation of basic geometric types



Figure 16. Generate NX CAD model from SysML block structure with geometric features

V. Application to the planned Europa Clipper Mission

Europa is one of the original four Galilean moons, meaning it was large and reflective enough to be seen through a primitive telescope. It is the second closest such moon to Jupiter, with Io being the closest. This is significant because Jupiter exercises a tidal flexing on Europa that generates sufficient heat to keep a large ocean of water liquid underneath a surface shell of water ice. In addition, Io's sulfur-expelling volcanoes leave behind sulfur-based compounds that may get transported into this ocean and provide components for organic molecules. For these reasons, the potential of this moon to harbor life is an intriguing scientific question.

A proposed Europa mission²³ currently in work would send a highly capable scientific spacecraft to the Jupiter system in order to understand the extent to which life is possible in that water ocean. In order to avoid the harsh radiation belts of Jupiter, the spacecraft would be sent on wide orbits that fly past Europa, but dwell far away from Jupiter. Each flyby would be highly dynamic and have multiple instruments performing remote sensing. An accurate understanding of the pointing of these instruments is necessary for understanding the science value of the mission.

Understanding the pointing of the instruments at a given time requires the synthesis of many pieces of information: where the spacecraft is at a given time, where it is pointed, where the individual instrument is mounted, and how it is pointed relative to its own degrees of freedom. This information is of course developed and distributed by different groups within the project.

In addition, a key problem is the simultaneous satisfaction of geometric constraints at all times in the mission. The definitions of what can articulate and in which way, as well as constraints on articulations, may be expressed across multiple models. In addition, many of these factors also figure into the pointing error budget of the spacecraft. The ability to coordinate the various pieces of information is required in order to get a coherent picture.

The systems engineering organization is responsible for coordinating this information. The Syndeia tool provides an excellent support for this coordination. Syndeia can reach into the NX model and obtain key information, including the two main elements of interest for this use case. The first is the rotation of a given NX part relative to a larger assembly within NX. The second is the key geometry information, such as a datum axis along the instrument's boresight. Both of these can be extracted from NX and brought into the system model. To understand the synergy between mechanical model information and system model information, a flowchart is shown below in Figure 17.



Figure 17. Key information in system and mechanical models and their synthesis

15 American Institute of Aeronautics and Astronautics

A. Scenario 1: Coordination of Error Budget Terms

The typical error budget for pointing combines elements from the Guidance, Navigation, and Control (GNC) subsystem; instrument mechanical design; structural dynamics like flexing; and spacecraft mechanical design.

Many of the terms in the error budget become requirements, and would be reasonable items to keep in the system model. In order to see the combined effects of mechanical misalignment (since they can stack up depending on geometry), the mechanical CAD is often most efficient.

B. Scenario 2: Rigorous separation of what-if, configuration, and real parts

One common practice in mechanical design is to place multiple copies of the same part in the CAD model. These copies may represent extremes in the range of motion, alternative positions as deployed or stowed geometry, or even alternative locations when the design is not yet fixed.

While the mechanical designer knows the difference between these uses, no one else looking at the model does. The system model is built for these kinds of distinctions and semantics, and thus makes a good support. The interaction between the two tools can work in a couple of different ways. The first is for the system model to have a representation of a part and mark it as one of these uses. This would be having the system model clarify CAD model semantics. The other possible use would be for the systems engineering team to develop a model that requests certain views be made in the CAD model. In this case, the system model seeds the CAD model or marks work yet to go.

Another benefit of this definition between the system model and CAD model is that the system model can then store parameters to be used later in analysis for matching worst-case scenarios or cleanly bookkeep values for different configurations.

C. Scenario 3: Coordination of multiple analyses

One of the prime uses for a system model is to provide the connective tissue between multiple domain-specific models. The system model is not as detailed nor as specialized as any of the typical models used on a project. However, it does provide the full mission context, which allows it to support translation between different teams.

The pointing analysis of instruments and other key components is a key example. The mechanical CAD model is employed to generate graphics showing the fields of view, keep-out zones, ranges of motion, and so on for different instruments. This information is also needed by analyses that schedule and analyze major mission activities.

The concurrent pointing of instruments and the solar arrays is critical to providing enough energy to sustain an active science flyby. The extraction of solar array information from an earlier version of the Europa CAD model (Figure 19) using Syndeia is shown in Figure 18. This information should be taken as representative only; even in basic layout it was only a notional, possible Europa spacecraft configuration.

The first thing to note in Figure 18 is that geometric information (location, rotation) is expressed as being intrinsic to the Association between two components rather than a property of any of the components. The Association couches this relationship as the geometric location of the component on the arrow side of the Association in the frame of the other component. Each component itself has information on physical extent (bounding box) as well as mass, volume, and moment of inertia. One key in using this information wisely is to realize that only a carefully modeled component will have accurate masses. The example below is not modeled in such a way. Syndeia allows for the orientation and location information to be pulled out of the mechanical CAD model. When combined with requirements, this information can be used to give articulation constraints to the mission planning team. Also, the technical parameters of key hardware can be gathered in the system model and be delivered directly to the mission planning and mechanical design teams concurrently. This digital connection between the mechanical design team and the mission planning team is a perfect example of model-based systems engineering. The systems engineering function of reconciling multiple data sources is directly supported by the system model. This is in opposition to adjudication through a stack of documents or series of meetings.

eblocka eNX Parts		
EHM_CLIPPER-SC-ASSY_SOLAR/A2		
Datum Coordinate System(0) : Datum Coordinate System(0)		
VIONICS_MOD-07262013_ASM692061: EHM_AVIONICS-MODULE_SOLAR/A2 EHM_PROPULSION-MODULE_SOLAR: EHM_PROPULSION-MODULE_SOLAR/A2	eblocks EHM_JUNO-SA-2-7X3	
mass: Real = 35941.38270730546 values	eparticipants eHM_CLIPPER-SC-ASSY_SOLAR/A2 EHM_CLIPPER-SC-ASSY_SOLAR/A2(end = eparticipants eHM_JUNO-SA-2-7x3/A1 EHM_JUNO-SA-2-7x3/A1(end = EHM_JUNO-SA-2-7x3)	}
density : Real = 5.543072380683069E-6 moments of intertia : String = 6.837951676410651E10,5.57660009628484E10,5.9413632848189514E10	values translation_vector_x : Real = 1.5916157281026244E-11	1
volume : Real = 6.48401829147257859 center of mass : String = -4.0587609956152,-6.132700537110424,2224.1773996633533 rinht upper corner x: Real = 2000.000112575753	translation_vector_y: Real = -5.4008353345125215E-11 translation_vector_z: Real = 508.0000000001751 rotation_matrix_px: Real = 1.0	
right_upper_corner_y: Real = -617.7076321911256 left_lower_corner_x: Real = -1821.40276625339566	rotation_matrix_zx: Real = 0.0 rotation_matrix_zz: Real = 0.0 rotation_matrix_zz: Real = 0.0	
left_lower_corner_z: Real = 3006/8533657248 right_upper_corner_z: Real = 5979.678799999848	rotation_matrix_zz : Real = 1.0 rotation_matrix_zy : Real = 0.0	
	rotation_matrk_yz : Real = 0.0 rotation_matrk_yx : Real = 0.0	
EHM_JUNO-SA-2-7X3	right_upper_comer_x : Real = 1074.9720088338286 right_upper_comer_y : Real = 2000.000112575753 bit hover comer_y : Real = 1074.072102862335	
EHM_JUNO-SA-2-7X3	left_lower_corner_y: Real = -2006.4985834887248 left_lower_corner_z: Real = 370.4998509094894	
eblock» eNX_Parts	right_upper_corner_z : Real = 3628.886732428523 «participants r : Real(end = translation_vector_x) «participants r I : Real(end = translation_vector_x)	
EHM_JUNO-SA-2-7x3/A1	<pre>«participant» r2: Real(end = translation_vector_z) «participant» r2: Real(end = translation_vector_z) «participant» r3: Real(end = rotation_matrix_xx)</pre>	I
Datum Coordinate System(1) : Datum Coordinate System(1)	<pre>«participant» r4 : Real(end = rotation_matrix_zx) «participant» r5 : Real(end = rotation_matrix_xz)</pre>	
MDH-EUROPA-SA-STRUTS-2014-06-30 : mdh-Europa-SA-Struts-2014-06-30/A1 EHM_JUNO_2-7X3-ASSY_RT : EHM_JUNO_2-7x3-ASSY_RT/A1	<pre>eparticipants r6 : Real(end = rotation_matrix_xy) eparticipants r7 : Real(end = rotation_matrix_zz) enarticipants r8 : Real(end = rotation_matrix_zz) enarticipants r8 : Real(end = rotation_matrix_zz)</pre>	l
EHM_JUNO_2-7X3-ASSY_LT : EHM_JUNO_2-7X3-ASSY_LT/A1	«participant» r9 : Real(end = rotation_matrix_yz) «participant» r10 : Real(end = rotation_matrix_yz)	I
mass : Real = 27380.57351318287 area : Real = 1.557360588121099E8	<pre>«participant» r11 : Real(end = rotation_matrix_yx) «participant» r12 : Real(end = right_upper_corner_x)</pre>	
 Idensity: Real = 7.795282990852571E-6 moments of intertia : String = 5.476716066345655E10,4.265969103889007E10,5.711716995765337E volume : Real = 3.51245408525270359 	<pre>sparticipants r13: Real(end = right_upper_corner_v) sparticipants r14: Real(end = left_lower_corner_x) sparticipants r15: Real(end = left_lower_corner_v)</pre>	
center of mass : String = -1.3475870523307647,2.8261378940477155,1694.4308419665263 right_upper_corner_x : Real = 2000.000112575807	<pre>«participants r16 : Real(end = left_lower_corner_z) «participants r17 : Real(end = right_upper_corner_z)</pre>	
right_upper_corner_y: Real = -137.50014909064623 left_lower_corner_x: Real = -1074.9720128562494 left_lower_corner_x: Real = 1074.9720128582492		
left_lower_corner_z : Real = -2006.4965346671 right_upper_corner_z : Real = 3320.886731092435		

Figure 18. SysML block structure generated for the Europa Clipper assembly (using Syndeia), showing solar array placement in the Clipper reference frame as a SysML association block



Figure 19. Europa Clipper Concept CAD model (circa 2015)

VI. Summary

In this paper, we have presented the concept of a Total System Model as the digital blueprint of a system, developed by combining the system architecture model (SysML) with models managed in a variety of disciplines, such as CAD, CAE, PLM, ALM, project management, requirements management, and simulation. This paper has focused primarily on the interface between systems engineering and mechanical/electrical design. We have presented the challenges in transitioning both systems engineering and mechanical design to a model-based approach and presented five use cases for integrating the two disciplines. We have also presented the Syndeia software platform and elaborated its capabilities towards addressing the five key use cases. Finally, we have presented the application of Syndeia and the concepts developed in this work to the planned Europa Clipper Mission being developed at NASA JPL.

VII. Lessons Learned and Future Work

In this section, we present some of the key lessons learned and potential areas of future work for this initiative.

- 1. Different abstractions in SysML and CAD modeling System engineers using SysML to represent the system architecture use different abstractions, such as (a) blocks and instances for representing an architecture pattern and creating instances, (b) generalization relationships for representing system variants and variation points, and (c) state machines to represent system states. On the other hand, CAD modeling environments use structural abstractions such as parts, assemblies, and features. There is no standards-based way to map the concept of blocks and instances to a CAD model since there is no equivalent concept of a CAD assembly template and instantiations of the template for specific assemblies. Parameterized CAD assemblies with resulting CAD models for different parameter values is an initial approach in this direction. Similarly, the concept of generalization is also not evident in the CAD modeling domain.
- **2. File-based storage of models** SysML models and CAD models are often stored as files in a local/shared file system or a version-controlled repository (e.g. PLM system). The file is the wrapper entity for the model. It is desired to store the system architecture model and the CAD model in an object-oriented or graph database to allow for scalability and versioning at the element level.
- **3.** Services and APIs to access system architecture and CAD models Access to the SysML model and the CAD model information requires the installation of tools on the system engineer's computer. It is desired that the models be available as a service allowing anyone in the organization with the right permission levels to access the information and connect/subscribe to it.
- **4. Ownership of models, visibility and permissions** A social aspect of integrating system architecture models and CAD models is the ownership of data and models. Technically, the ability to synchronize CAD model parameters to the system architecture model, and updating part/feature parameters in the CAD model from the SysML model, is available in Syndeia. However, system architects and mechanical designers may prefer to have a buffer zone to receive the change requests, evaluate the impact of the change in the system architecture and CAD modeling environments, and then approve the change before effecting it. There is also a new balancing act to be had among the teams between simply opening up models without guidance and requiring a gatekeeper to ask for any piece of information.
- **5. Non-parameterized CAD models** One of the biggest roadblocks to fine-grained interoperability between system architecture and CAD models is that the latter is often not a parameterized model. For enabling true model-based engineering, we need to incentivize the mechanical design teams to produce parameterized CAD models based on system specifications.

Acknowledgments

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

We would also like to acknowledge Rose Yntema, Anshu Dwivedi, Amit Kumar, Alex Phung, and Manoj Waikar at Intercax for their significant contributions to developing and maintaining the Syndeia platform.

References

¹International Council on Systems Engineering (INCOSE). "Systems Engineering Vision 2020". Version 2.03, TP-2004-004-02. September 2007

²Friedenthal, S., Moore, A., Steiner, R. "A Practical Guide to SysML: The Systems Modeling Language". Third Edition. The MK/OMG Press. Elsevier Science.

³Systems Modeling Language (SysML), version 1.4, Object Management Group, <u>www.omgsysml.org</u>

⁴Syndeia 3.0 (Intercax), <u>www.intercax.com/syndeia</u>, accessed Aug 17, 2016

⁵Git, <u>https://git-scm.com/</u>, accessed Aug 17, 2016

⁶Cole, B. and Jenkins, S.J. "Connecting Requirements to Architecture and Analysis via Model-Based Systems Engineering", AIAA Infotech @ Aerospace, AIAA SciTech, (AIAA 2015-1116), <u>http://dx.doi.org/10.2514/6.2015-1116</u>

⁷SysML 1.5 Property-based Requirements Working Group, <u>http://www.omg.org/members/sysml-rtf-</u>wiki/doku.php?id=rtf5:groups:require:requirements

⁸Teamcenter (Siemens PLM), <u>http://www.plm.automation.siemens.com/en_us/products/teamcenter/</u>, accessed Aug 17, 2016

⁹Windchill (PTC), <u>http://www.ptc.com/product-lifecycle-management/windchill</u>, accessed Aug 17, 2016

¹⁰NX (Siemens PLM), <u>http://www.plm.automation.siemens.com/en_us/products/nx/</u>, accessed Aug 17, 2016

¹¹Creo (PTC), http://www.ptc.com/cad/creo, accessed Aug 17, 2016

¹²GitHub, https://github.com/, accessed Aug 17, 2016

¹³JIRA (Atlassian), <u>https://www.atlassian.com/software/jira</u>, accessed Aug 17, 2016

¹⁴DOORS-NG (IBM), <u>https://jazz.net/products/rational-doors-next-generation/</u>, accessed Aug 17, 2016

¹⁵Mathematica (Wolfram Research), <u>http://www.wolfram.com/mathematica/</u>, accessed Aug 17, 2016

¹⁶Simulink (MathWorks), <u>http://www.mathworks.com/products/simulink/</u>, accessed Aug 17, 2016

¹⁷MySQL (Oracle), http://www.mysql.com/, accessed Aug 17, 2016

¹⁸Excel (Microsoft), <u>https://products.office.com/en-us/excel</u>, accessed Aug 17, 2016

¹⁹Bajaj, M., Zwemer, D., Peak, R., Phung, A., Scott, A., Wilson, M. (2011). "Satellites to Supply Chains, Energy to Finance — SLIM for Model-Based Systems Engineering, Part 1: Motivation and Concept of SLIM". 21st Annual INCOSE International Symposium, Denver, CO, June 20-23, 2011. <u>http://www.omgsysml.org/SLIM for MBSE Bajaj Part1.pdf</u>

²⁰Bajaj, M., Zwemer, D., Peak, R., Phung, A., Scott, A., Wilson, M. (2011). "Satellites to Supply Chains, Energy to Finance — SLIM for Model-Based Systems Engineering, Part 2: Applications of SLIM". 21st Annual INCOSE International Symposium, Denver, CO, June 20-23, 2011. <u>http://www.omgsysml.org/SLIM for MBSE Bajaj Part2.pdf</u>

²¹Bajaj, M., Zwemer, D., Phung, A., Yntema, R., Kumar, A., Waikar, M., Dwivedi, A. "MBSE++ — Foundations for Extended Model-Based Systems Engineering Across System Lifecycle". 26th Annual INCOSE International Symposium, Edinburgh, Scotland, UK, Jul 18-21, 2016.

²²Piper Cub J3 Plane Creo model (GRABCAD), <u>https://grabcad.com/library/piper-cub-j3-1</u>, accessed Aug 17, 2016

²³JPL Europa Mission, <u>http://www.jpl.nasa.gov/missions/europa-mission/</u>, accessed Aug 17, 2016