

# A Model-Based Approach to Developing Your Mission Operations System

Robert R. Smith<sup>1</sup>, Kathryn A. Schimmels<sup>2</sup>, Patricia D Lock<sup>3</sup>, and Charlene P. Valerio<sup>4</sup>  
*Jet Propulsion Laboratory / California Institute of Technology, Pasadena, CA, 91109*

**Model-Based System Engineering (MBSE) is an increasingly popular methodology for designing complex engineering systems. As the use of MBSE has grown, it has begun to be applied to systems that are less hardware-based and more people- and process-based. We describe our approach to incorporating MBSE as a way to streamline development, and how to build a model consisting of core resources, such as requirements and interfaces, that can be adapted and used by new and upcoming projects. By comparing traditional Mission Operations System (MOS) system engineering with an MOS designed via a model, we will demonstrate the benefits to be obtained by incorporating MBSE in system engineering design processes.**

## I. Introduction

System engineering is essential to the design of complex systems. It provides a multidisciplinary approach to both the technical and management needs of in system design. As modern systems continue to increase in complexity, more rigorous and standardized practices are needed. The use of Model-Based Systems Engineering is an emerging paradigm set to meet this need.

MBSE helps to manage complexity by moving systems engineering practice from an approach that is primarily document-based to one that is model-based. In this paradigm, a core model represents the collaboration of many systems, and in turn, becomes the basis of a project's artifacts such as requirements, design specifications, and verification information. This contrasts with the traditional method of capturing system design details in a number of different documents, leading to problems with completeness, traceability and understanding. The Jet Propulsion Laboratory (JPL) has extended MBSE to the design of a mission operations system, which consists of a ground-based operations system and the personnel, processes, and procedures needed to achieve mission success. Traditionally, an MOS is developed through adaptation of previous systems. However, this has proven to lead to serious limitations and inefficiencies, historically impacting missions in terms of time, money, and risk. An MOS built via a model, while still observing proper systems engineering practices, provides benefits to designers such as complete capture of the system design, a single authoritative source of information, design artifact generation, lessons learned capture, flight-ground trade analysis, and savings in time and risk.

## II. How Do We Currently Build Our Mission Operations System?

Development of systems through a document-based approach is a time-proven method. In the current document-based paradigm, the Mission Operations System Engineer (MOSE) briefly analyzes a new mission's objectives, high-level requirements, architecture, and constraints. Based on that analysis, the MOSE determines a similar mission's MOS that could be adapted to the new mission, and collects its MOS design artifacts. The MOSE and a Ground Data System Engineer (GDSE) scrutinize the heritage design for differences, lacks, and unneeded capabilities, and based on the heritage design, create a draft design for the new mission. This design is analyzed

---

<sup>1</sup> Manager, Operations Revitalization, Mission Operations System Engineering Group, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109, MS 301-270, Member.

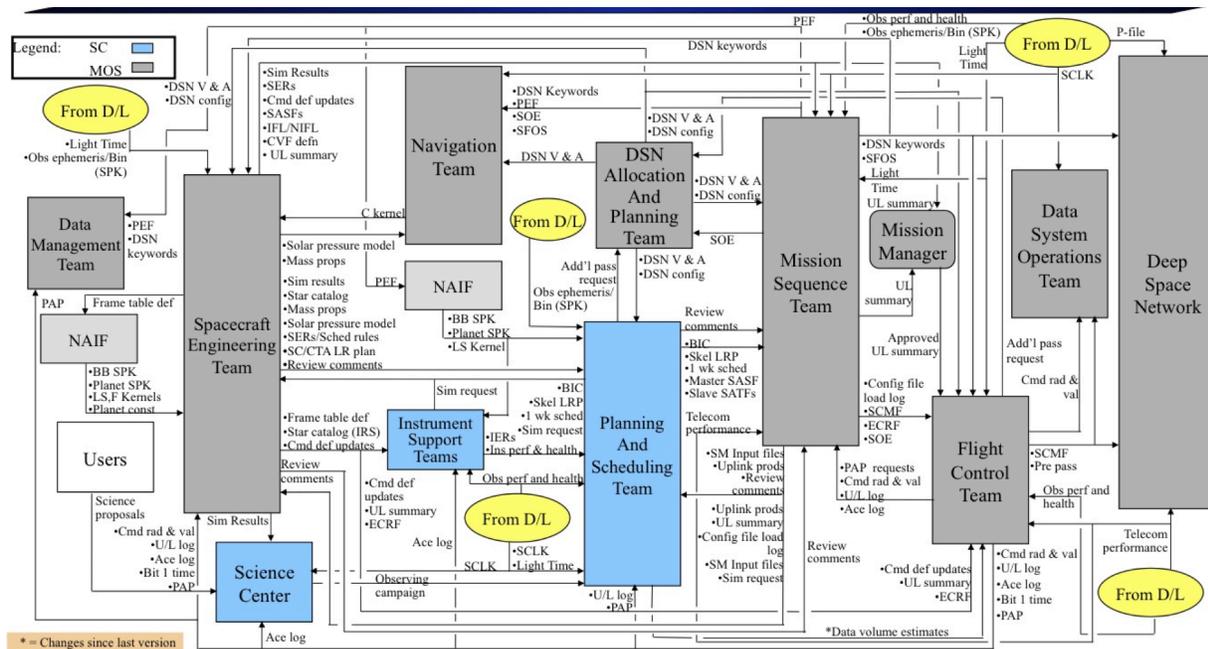
<sup>2</sup> Europa Clipper Mission Operations System Engineer, Mission Operations System Engineering Group, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109, MS 321-560, Member.

<sup>3</sup> Mission Operations System Engineering Group Supervisor, Mission Systems Engineering Section, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109, MS 301-270, Member.

<sup>4</sup> STABLE Mission Operations Systems Engineer, Ground Data System Engineering Group, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109, MS 301-270, Member.

further to ensure that it meets the needs of the new mission. Over the early development period, subsequent analyses and trades occur until a preliminary design for the new mission is considered complete.

Meanwhile, spacecraft and instrument design has been taking place ahead of MOS design, which is often begun six months to a year later than flight system design. Traditionally, the flight and ground systems have been treated as if their designs interact only at the flight-to-ground interface. Because the flight system is maturing ahead of the ground system, interactions between the two systems may be overlooked without input from mission operations. During the preliminary and critical design phases, the MOSE must work to adapt the previous MOS design to the new mission as well as impact any previous decisions made in flight system design, and monitor for changes that can affect the operations system.



**Figure 1. Heritage Mission Operations System Information Exchange Overview**

The traditional design paradigm also has limitations that impact development and operations risk, some of which are illustrated in Figure 1. Functional elements in this system are highly interwoven, each element tied to most of the others. If new functionality is added to the system, the complexity of the interactions increases geometrically. Integration of the overall system is arduous, requiring months of testing, and regression testing for minor changes can take weeks. Functionality is provided on a somewhat ad-hoc basis, based more on team structure or software implementation than on what the *system* needs to do.

There is a need for a way to capture the complexity of each component, describe how it interacts with the other system components, and capture the functionality of both the component and the overall system. Diagrams and terms need to be represented in a common manner and refer to items using consistent names.

The weaknesses in the current MOS design process have led system engineers to transition to a model-based approach, controlling the *model* of the MOS instead of its documentation.

### III. Why Model An MOS?

Two key architectural principles drive the application of an MBSE approach to the development of a mission operations system model. First, the MOS model must be treated as a control system that is capable of “closing the loop” on its required objectives, that is, the performance of any planned activity is also analyzed based on that plan. Second, the MOS model must support a “develop with what you fly with” paradigm, meaning that the system used to develop a mission is also the system that operates the mission<sup>1</sup>. Applying these architectural principles results in a model consisting of elements that represent the system’s requirements, design, rationale, and interfaces. Further, Systems Modeling Language notation (SysML) provides a standardized language for representing aspects of the

model in terms of its requirements, structure, and behavior, thus allowing it to be viewed from different subsystem perspectives.

In addition to SysML's standardization benefits, the model itself represents a single source of truth, i.e., a sole repository of information that can be version controlled. An MOS model enhances traceability among use cases, requirements, and implemented functions. Lines of responsibility and authority over information become clear when a model provides an unambiguous design of the system's composition, functionality, and information flow. Modeling also offers some automation capability and supports syntax verification, enabling verification of the model itself as well as allowing early design verification. As the model consists of reusable and adaptable products based on the knowledge of MOS subject matter experts, it also serves as a powerful tool for knowledge capture, leading to better cross-system understanding and mission-to-mission experience. That experience, captured in the model, allows an MOSE to quickly identify interfaces and standard mission elements, and therefore focus their attention on the unique aspects of the new mission. This helps MOSEs to even out the flight and ground system development processes when MOS development begins later.

A significant benefit comes from the ability to produce standard system description documentation, e.g., gate products<sup>2</sup>, directly from the model. By maintaining a system's information and relationships within the model, changes to documentation are dynamically captured as design changes are made. Creating documents, a task that can take weeks, becomes a much-simplified effort when they are instead *generated* from the model, saving time while mitigating risk. At scheduled points in the development process, documentation is produced by populating existing templates with the design information available from the model. In a similar way, reviews and review packages are improved by modeling. Document-based review package pictorial diagrams often contain abstractions of system information or misrepresent some aspect of the system. Review boards frequently find discrepancies between the information presented by different subsystems. Unlike pictorial representations, model views depict the system's information consistently and accurately, regardless of a subsystem's unique perspective. Thus, model views change dynamically as the system's design matures.

It must be noted that building a model is not a replacement for systems engineering. Rather, a model-based approach should be viewed as a tool to aid the system engineering effort across the full MOS. Instead of an MOS design retooling past or existing missions, modeling allows engineers to take a fresh look at what the system needs to accomplish.

#### **IV. Traditional MOS Design vs. MOS Designed via a Model**

In addition to the two key architecture principles introduced above (close the loop, develop with what you fly with), the use of a model-based approach enables several additional architectural principles to be realized. The first principle is to build common solutions to common problems via a single authoritative source of information and the second is to learn from experience, by incorporating lessons into a re-usable modular MOS design.

We've identified five key challenges that a system engineer faces in the traditional MOS design methodology: maintaining a single, authoritative source of design information; requirements development and verification; impact assessment in flight/ground trades; capture of operational interfaces and agreements; and the transition of knowledge and information from development to operations. These challenges and the benefits of using a model-based MOS design approach are explained below.

##### **A. Single Authoritative Source of Design information**

A primary task in development is to capture the design as it evolves. In the early and mid-design phases, the MOSE's challenge is working with designs that are constantly changing and linked quite closely to changes elsewhere in the larger mission or project system. Text documents and presentations have traditionally been used to record and communicate design information. This approach is time consuming, suffers from redundancy and overlap of information among multiple sources, and has no single authoritative source. As such, the information is often out-of-date, and therefore ignored by the intended audience.

The focus of the development systems engineering process should be on identifying and capturing the complete set of MOS system components, interfaces, internal and external agreements, operations processes, and procedures – and seeing them clearly connected. This requires a standard way to capture, document, and visualize the MOS design, in particular its connection to how the MOS controls the flight system. A central step in MOS development is identifying the operational scenarios for the flight and ground systems<sup>3</sup>. Scenarios drive the requirements, agreements, interfaces, and processes. Scenarios, however, are not static – they require iteration, collaboration, and are very fluid early in the lifecycle. In order to capture and communicate them effectively, we need a single source

for scenario documentation and information. Using a model-based system to do so reduces the risks brought by multiple versions of repetitive or conflicting information.

To capture scenarios effectively, system engineers need to be able to:

- Incorporate both text and diagrams to communicate intent,
- Allow collaboration - multiple authors must edit and review content in a configuration-controlled manner,
- Easily export scenario documentation; more specifically, generate an up-to-date document at any point,
- Easily and explicitly link requirements to scenarios, in support of verification and validation (V&V),
- Use scenarios to identify and directly link to a list of principal interfaces or agreements early in the development lifecycle, and later link directly to the actual interface agreements.

Desired Features for Scenario Capture and Communication	Wiki	Online Scenario Database	OpsRev MOS Model
Collaborative authoring, editing, & reviewing	✓	✓	✓
Single authoritative source	Not always	✓	✓
Document Export & Generation	Copy and paste.	significant post-export editing required to incorporate into documents	<ul style="list-style-type: none"> <li>• Standard document templates</li> <li>• Post-export editing allowable but not required</li> <li>• links to online model management web-editing capability</li> </ul>
Requirements linkage	X	✓ (static)	✓ (dynamic)
MOS products integrated throughout project lifecycle	X	X	Scenarios tied to: <ul style="list-style-type: none"> <li>• Agreements &amp; Interfaces</li> <li>• Processes</li> <li>• Services, Teams, and Roles</li> </ul>
Validation and analysis capability	X	X	<ul style="list-style-type: none"> <li>• Scenario meets requirements</li> <li>• Interfaces used during scenario</li> <li>• Metrics to check completeness/correctness</li> <li>• Link to analysis tools</li> </ul>

**Figure 2. A comparison of several methods for capturing scenario information.**

Several approaches have been used on flight project development efforts to capture and communicate scenarios – from the traditional viewgraph charts and text descriptions, to wiki pages, to an online repository database. Figure 2 compares three methods for scenario capture. The model-based approach offers solutions across all of the desired areas:

- Provides a model repository - Single authoritative source for all information,
- Allows collaboration among users for editing and reviewing scenarios,
- Allows collaboration among users for editing and reviewing scenarios,
- Up-to-date documents are easy to generate and maintain,
- Gate product templates (e.g. Operations Concept Document) are easily populated and exported in a variety of formats,
- Scenarios can be linked with requirements in the model,
- The model persists throughout the lifecycle allowing scenarios to be tied to agreements, interfaces, and processes all in one place,
- Analysis tools can validate that scenarios meet requirements (also allows use of external tools),
- Provides the capability to link scenarios to the teams, roles, and processes needed to carry them out,
- Provides metrics on the completeness and correctness of the model.

The use of a model-based approach early in the development process reduces the time spent documenting design information, and reduces inconsistencies introduced by multiple sources of information, thereby reducing development risk.

## **B. Developing and Verifying Requirements**

The requirements development and verification process has its own challenges. Key is identifying the right set of requirements applicable to the mission under development, and avoiding inappropriate items from previous projects. It is also necessary to provide a clear mapping of the requirements to operations scenarios and the processes that respond to or fulfill them. Verifying requirements against the design early in the lifecycle mitigates costly fixes late in development.

Requirements must be derived from the operations concept, scenarios, and architecture-driven interfaces early in the development of the operations concept. Typically, the mapping of requirements to test and analysis plans is done via a requirements management system such as DOORS, and is not connected to the operations scenarios and processes that are to be validated. Additionally, requirements change frequently throughout the development lifecycle, even at the V&V stage. Whereas a requirements management tool can provide linking between hierarchical levels of requirements and verification plans and status, it lacks the ability to interrogate the source of and resulting design from the requirement. In the case of an MOS development effort, knowing where the connections to the operations processes, interfaces, scenarios, and external agreements are allows clearer understanding of the impacts of design changes and trades.

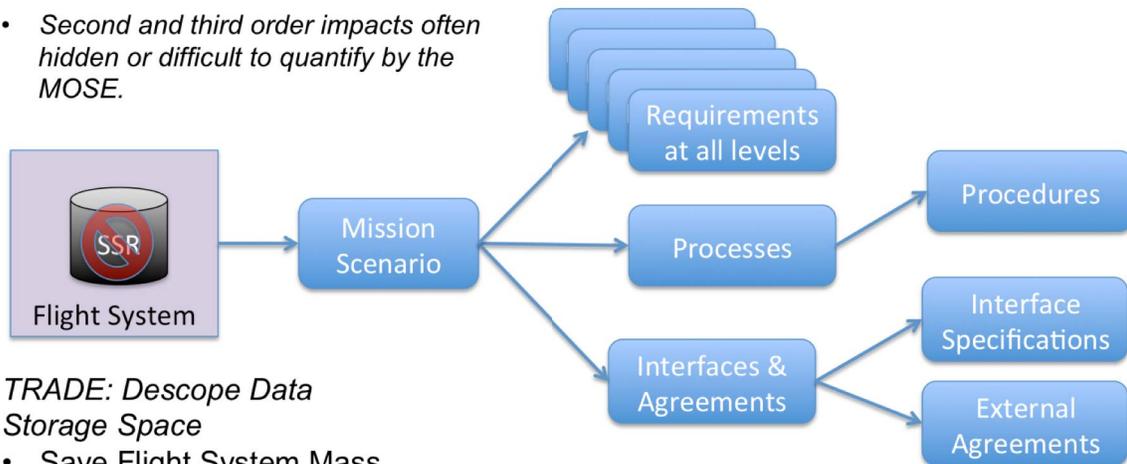
Using a model-based system to capture requirements not only provides the traceability of a typical requirements management tool, but also adds the ability to specify physical, interrogatable connections between elements in the model. It allows the system engineer to assess completeness and correctness of a requirement, and can include syntax review to identify poorly written requirements. Using this approach, requirements will be written correctly in the early stages of development, reducing the need for re-work due to poorly written “shall” statements.

## **C. Flight / Ground Trade Assessment**

As the flight and ground system designs proceed, trades are performed on function ownership. Should certain functionality be provided onboard the flight system, or in the ground system and MOS? Is this a scope increase, and if so, to which system? Every major operability trade has an impact on the flight system as well as the ground system. Often, though, impacts to the flight system are simple to characterize in terms of a consumable or margin impact (e.g., mass, power, data volume, etc.), or development budget and schedule. Changes to flight system capabilities, however, often drive additional requirements and costs to the ground system, and the impact may be harder to assess qualitatively without complex analyses. Figure 3 illustrates the difficulty in assessing all of the impacts in an example mission trade. Using a model to assess the real, quantifiable impact of proposed flight system changes is essential to performing balanced trades with equal options to consider.

One way of enabling sound trade studies is validation of flight operations scenarios early in the development lifecycle. In order to do this, we need reliable models to capture and analyze scenarios. Performing this validation early in the lifecycle reduce the impacts when, during system-level scenario testing, a pivotal scenario turns out to be faulty. Early modeling and analysis reduces re-design, work-arounds, additional costs, and risk late in development. The essential method, provided by the MOS model-based approach, is to close the loop early and ask – does the system or approach do what was intended? System engineers use the model, paired directly with analysis tools, to identify the impacts to operability in design trades.

- Second and third order impacts often hidden or difficult to quantify by the MOSE.



*TRADE: Descope Data Storage Space*

- Save Flight System Mass
- Save Flight System Cost
- Resolve Fit issues on S/C Bus

*TRADE: Descope Data Storage Space*

- Increase in ground station passes → costs
- Decrease in data collection volume/day
- Increase instrument contention for data collection time
- Longer sequences, change to seq dev process/timeline
- Increase in science mission duration
- **But, what else are we missing?**

**Figure 3. A typical flight ground trade: reducing/removing onboard data storage has a number of unanticipated impacts to operations scenarios, processes, and interfaces. A good model can help to analyze and identify the ripple effects of such a change.**

#### D. Capturing Operational Interfaces and Agreements

A large part of MOS design includes the definition of interfaces for information flow, and agreements among organizations and team members. The mission operations system is unique in that it is the only system in an interplanetary mission that includes personnel as one of its vital elements. Hence, it is imperative that we capture interfaces and agreements in a way that is consistent and follows a standard, and that can easily be updated when system behavior changes.

Identifying, documenting, and validating interfaces and agreements is a laborious task requiring not only to be able to identify *what* the interface is, but also *why* the interface is needed. Poorly designed systems define interfaces without the understanding of why, when, and how the data are to be used.

The benefit of model-based systems engineering over traditional document-based design capture in this context is expansive. 1) It provides a view into the complete set of internal and external agreements between the MOS and its peer systems (ground communication networks, data archive systems, etc.), and the explicit specifications of the interaction – timeliness, frequency, content, etc. 2) It provides a standardized method to capture specifications for all interfaces. A viewpoint can be generated in the model to capture and convey the appropriate information in a standard way for all missions. The content has the ability to vary greatly, but the type of information captured and reported can be standardized. 3) A model provides a single authoritative source to go to for the latest revisions and state of the interfaces and agreements, providing version control and robust configuration management. The latest revision of information is always clearly available, as well as knowledge of the state of the agreement - i.e. when all parties have signed an agreement or specification that is ready for testing and operations. 4) The model provides the ability to tie the interfaces directly to the teams, roles, and processes in which they are exercised. This makes interface verification and validation easier to track during process thread testing and operational readiness testing, as well as aiding in training exercises. 5) The model also allows the capability to generate reports and metrics to analyze the completeness and correctness of the interfaces, and to identify missing agreements.

## **E. Transition of Information and Knowledge into Flight Operations**

One goal of a Mission Operations System Engineer is to put in place for the operations team a system that will allow the team to easily align ground plans with any changes that occur in flight system operations. The use of a model allows quick understanding of impacts to MOS functions due to a change in how the flight system is operating (e.g. anomaly impacts to MOS design). It is also crucial for the system to allow the team to assess proposed changes, providing the ability to see how a proposed change would impact how the flight team operates.

Another vital area is the transition of development knowledge, products, and information into operations. Traditionally, missions experience a lack of spacecraft & subsystem engineer availability to capture “how to operate the subsystem” information in the rush leading up to launch. Typically, knowledge is transferred by email, potentially out-of-date design documents, and incomplete or immature operations procedures. This incomplete information then weakens operations team training.

An MOS model alleviates these problems by providing useful products for training and operational readiness testing early on. From within the MOS model, teams are able to “fly the mission” thousands of times before ever launching. The information needed by operators for procedures and anomaly investigation has been captured, and persists in the model throughout the mission duration. Primary system design information is more easily transitioned from the development team to the operations team, with less chance of “missing something”. *Persistence of a model through the entire lifecycle is the key!*

## **V. How Do We Build Models?**

The approach to designing a mission operations system using model-based systems engineering is not that different than for any other system engineering task. The first step is to define the scope of the task, which includes defining the system boundary, needed information, level of detail, and the documents to be produced. The scope of the system information can include use cases, system requirements, system composition, interfaces, processes, system state, information products, and relationships between system information (such as relating planned states to actual values). Each of these information products has multiple levels of detail that can be specified. The details of the scope become the metrics against which progress of the system engineering task is measured.

To put together a modeling team, three areas of expertise have been identified as key. First is expertise in the underlying model-based enabling engineering platform. This includes the modeling software, collaborative model information repository, document-generating software, and document server design and maintenance. The second area of expertise is an understanding of the modeling architecture framework. The framework includes the standard model representation for each piece of information to be modeled. This expert is also responsible for the proper use of model-based system engineering in the design of the system. The third area consists of the mission operations system engineers and subsystem domain experts who are responsible for the overall design of the mission operations system. In modeling the system, the mission operations system engineers must *build the right system*, while the model engineers must *build the system right*.

The diversity of the team leads next to the diversity of stakeholders and reviews. Model progress and products are reviewed for technical content, proper use of modeling methods, and performance. Stakeholders in each domain area review the model-generated information for accuracy and completeness. As the modeling effort progresses, regular discussions are held with domain experts external to the task to facilitate dialog and engage experts with different experiences and opinions. Materials generated from the model are distributed for review to mission operations and subsystem engineers across a variety of missions. The use of customer advisory groups is also helpful in addressing models where each customer has a different view of the system design. Together as a group, a common solution can be identified. The model framework and its application require a different set of reviewers who are knowledgeable in model-based systems engineering and the associated best practices. Regular progress reviews evaluate schedule, risk, budget, staffing, and progress against established milestones and metrics.

A schedule is built based on the needed model components and the order in which those are to be built. The schedule follows a system engineering approach of sketching out a concept, identifying the requirements, then designing a system that meets the requirements. The model development lead identifies the model elements to be built, the documents to be generated from the model, and the model framework to be developed. Document reports are helpful in assessing the accuracy and completeness of the model. Either a traditional waterfall schedule or something similar to a software agile development schedule method is used. The agile schedule method can be more beneficial for new types of development where the schedule uncertainty is large as it allows breaking a large task into smaller, more measurable pieces of work. Short development cycles of approximately one month produce useable products on a regular basis. When unexpected problems arise, priorities can be readily shifted to allow other work to be completed ahead of schedule. Stakeholders can incrementally review generated material. Showing

progress against the complete set of items enables tracking of progress for the overall effort and will clearly show the work remaining.

In the past, mission operations engineers would inherit requirements from previous missions, identify those that apply to the new mission, and fold in any new requirements to match the new mission's needs. There are typically hundreds of mission operations system requirements that are needed, and depending on the expertise of the system engineer, some areas may be better understood than others.

Now using a model-based approach, the system engineer will identify a set of use cases for activities that the mission needs to perform. Each use case then specifies a number of requirements, as shown in Figure 4. The system engineer identifies the use cases that apply, and then accepts only those requirements that are applicable to the mission's specific needs. This enables employing a standard set of use cases tied to a reference set of requirements that are applicable to a wide variety of mission types. Missions adapt the requirements to meet their specific mission needs. Adaptation includes identifying specific timing needs and data rates.

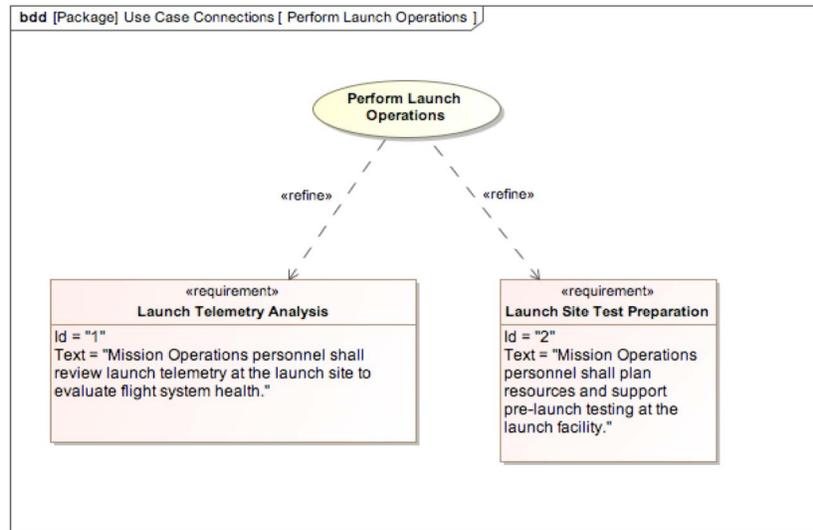


Figure 4. The Perform Launch Operations use case addresses requirements for analyzing launch telemetry and preparing for launch site tests.

Once requirements have been specified, the mission operations system composition is specified with a defined system scope. The composition includes the mission operations system and the subsystem components, which are referred to as services. The requirements constrain the system and services, and their functionality.

Each service is allocated responsibility for performing specific functions. The functions identify the planning, execution, and analysis processes that the service performs, as illustrated in Figure 5. Each service is designed to plan its activities, perform its tasks, and analyze its performance based on the plan. The analysis is fed back into the next planning cycle to infuse lessons learned and enhance performance and reliability.

Once functionality is identified, the information inputs and outputs are identified. Information exchanged with peer systems, those systems external to the mission operations system, is identified first. Next, interfaces between the internal services are identified. Inputs can be directional (control flow) or information products. Each service becomes the authoritative source for the products it produces and can send control information to another service. For each control flow sent to another system or service, a corresponding response is expected.

At this point, the system has all its inputs, functions, and outputs identified. Next, the processes need to be specified. For each service and each function, a process flow is specified at a high level. In modeling, best practices suggest keeping the number of activities to a small number of higher-level steps to control complexity. Higher-level processes are elaborated into lower level steps and where possible,

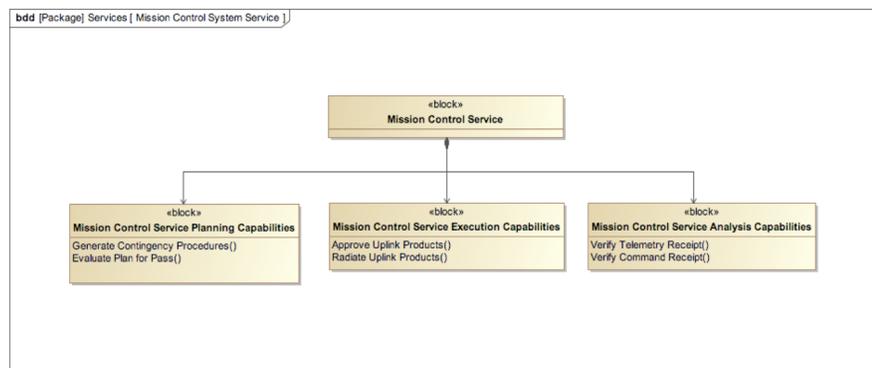


Figure 5. An example showing that the planning, execution, and analysis capabilities are part properties of the Mission Control Service.

and re-useable processes identified, e.g., each process does not uniquely define how to perform anomaly notification; instead, there is a common process for anomaly notification.

As processes are developed, there may be changes to information flows. This then becomes an iterative process of designing and refining the inputs, outputs, and processes. The development schedule must be planned to allow for some rework.

The various roles within the mission operations system are modeled as actors. A role represents the entity that performs a specific function. Typical roles include a mission planner, a command operator, and an attitude control engineer. When individuals with different backgrounds discuss roles, each person may have his or her own understanding of what each role does. Within the model, a role is only associated with a specific function or set of functions. This allows all team members to use the same terms when discussing system design, functionality, and the role responsible for producing a product.

Teams and roles have often been confused. In a model, a role performs a function, and a function can be performed by a person or by software, therefore a role can represent a person or software. Roles are explicitly defined and mapped to functions, or specific activities within the processes. Teams, on the other hand, are a collection of roles. For a reference model, a typical set of teams may be identified, but the mapping of teams to roles is left to the mission to decide, based on their preferences. A mission transitioning out of a high activity period such as launch and checkout may reduce staff by consolidating their workforce so that fewer teams perform more roles than during the high activity periods. The roles remain the same, and the functions do not change. Rather, the teams take on additional roles.

Once the requirements are linked to the services and their specific functions, and functions are tied to information and roles, traceability matrices are generated from the model. Verification and validation details are easily exported with the system design information and tests are designed around one or more views of the model. “Thread” tests are based on a flow of information, process steps performed by roles, and expected results of the process. Following this pattern, training plans for each role can be quickly extracted from the model.

A model-based approach leads a designer through the system engineering steps needed to develop a system that is designed based on the way the system will be used. Requirements are traceable to system components, the functions those components provide, and the roles performing the functions. Naming model elements forces the names of system items to be consistent across all services, with specific descriptions of each model element. Information flows are expressed explicitly, and any mismatches quickly become evident. For components that need to be updated or replaced, the functionality and information that needs to be replaced by the new system is clear. Each model element captures a specific aspect of the system design. Model-generated documentation is linked directly to the single authoritative source of design information. As the design evolves, the documentation automatically reflects the changes. This includes design description documentation, traceability matrices, and training material. From this list, one can see that using a model-based approach greatly enhances the MOSE’s ability to engineer the details of the system.

## **VI. What Models Have We Built?**

Using the methods detailed above, a number of models have been built under pilot programs, whose intent was to produce prototypes. Others are larger models intended to replace or augment existing capabilities.

One of the most detailed models that pioneered many modeling techniques is Operations Revitalization, or “OpsRev”. OpsRev is a subset of the MOS 2.0<sup>4,5</sup> initiative that is developing a next-generation operations system. MOS 2.0 is planned to incorporate a number of models in its tool suite, while also encapsulating operations practices that have too long been left to providence. Via models of the system, developers can use ground tools as soon as they are adapted to a minimum set of mission parameters. Design is supported by the same tools used in operations, and analyses and tests of flight-to-ground interactions performed at every stage. This yields time and risk savings over the previous paradigm, in which only some of the Phase D activities were performed using the operations ground system.

Another modeling project related directly to MOS 2.0 is the MOSE and GDSE development procedures model. Both the MOS and GDS development efforts employ a lead system engineer that is responsible for the overall development and delivery of the system and all of the necessary design artifacts. In order to correctly accomplish the many steps of these roles, procedures have been written to capture the details of the day-to-day development process and responsibilities. These procedures are routinely updated and have been maintained as text files for years. As a training exercise, the procedures were modeled using SysML and Business Process Modeling and Notation (BPMN). Though the two existing procedures were considered to be “very good” by organizational management, the activity of modeling them pointed out many areas where enhancements could be made and information capture

could be improved. Ambiguity was widespread, and the two procedures, which must be aligned, were inconsistent in a number of areas. Engineers with more than 20 years of experience in MOS and GDS were surprised to see how many these issues the modeling effort uncovered in these mature, straightforward processes.

Work in a related area has been done to model verification and validation (V&V) activities<sup>6</sup>. This was part of a pilot program to determine if static models could help with V&V work, and if so, could executable models be built to perform early requirements verification. These models complement MOS models, using the requirements and design model-generated products as input to verification planning. Inheritance from the design models themselves allows faster planning and checking of test procedures, which in turn can be used to generate test scripts. With an executable model, test scripts can be run against the model and provide early design verification. The pilot program proved the value of the effort, though necessarily, the limited design models of a pilot program yielded limited results. The program continues.

## **VII. Future work in MOS and the Ground System**

Part of the MOS is the ground data system (GDS), which performs a set of roles and functions within the MOS. The GDS is a system composed of software capabilities that operate on hardware platforms, and rely on infrastructure services, such as the ability to print or exchange information between software components. MOS processes and information interfaces describe what the GDS must do. Upcoming work will define a reference GDS model and associated views of the system, including describing how GDS software maps to operational processes.

The MOSE and GDSE development procedure models, described above, were completed as planned. However, the exercise provided many suggestions for improvements. Future work will connect the two system engineering procedures to the OpsRev model, streamlining system engineering workflow. More detail will also be added, and the model will capture research resources, “tips and tricks”, lessons learned, and historical contacts. Eventually, the combined MOSE/GDSE procedure model, in concert with OpsRev and the GDS model, will provide most of the tools necessary for a system engineer to efficiently design, build, test, and deliver an operations system to a flight project.

A number of efforts to model spacecraft and instrument systems are also underway. As the various independent models are developed, a parallel effort is underway to develop model interoperability, leading to better analyses and trade study outcomes. A number of deep space, Earth, and Mars missions have employed SysML models to capture design topics smaller than a full operations system. The next generation of JPL missions, most notably NASA’s Mars 2020 and Europa Clipper missions, have begun early work to fully model both the flight and ground systems throughout development and into operations.

## **VIII. Conclusion**

Transitioning from a traditional, document-based approach to mission operations system engineering to a model-based approach offers significant benefits. The traditional approach spreads requirements, design, and analysis information across several documents, limiting understanding of an MOS’s design to either a macro- or micro-view. MBSE serves to address these matters by building a model to serve as the single source of truth for communicating the design. This then allows unambiguous definitions of interfaces between systems and their information flows, and effective knowledge capture between missions.

In this paper we’ve shown that applying MBSE is *not* a change in systems engineering practice. Rather, it provides principles that allow system engineers the freedom to apply creative approaches to the design of an MOS while still observing commonalities essential to an MOS’s architecture. This approach to building an MOS model has successfully been put into application in the Operations Revitalization and MOSE and GDSE development models. As we move forward, we will continue to explore the positive impact that MBSE has on MOS design and system engineering.

## **Acknowledgments**

The work described in this paper was performed at the Jet Propulsion Laboratory (JPL), managed by The California Institute of Technology (Caltech), under contract to the National Aeronautics and Space Administration (NASA).

## References

<sup>1</sup>Bindschadler, D.L., Boyles, C.A., Carrion, C., and Delp, C.L., "MOS 2.0: The Next Generation in Mission Operations Systems," SpaceOps 2010 Conference Proceedings, Huntsville, AL, Apr 25-30, 2010.

<sup>2</sup>Jackson, M., Delp, C., Bindschadler, D., Sarrel, M., Wollaeger, R., Lam, D., "Dynamic gate product and artifact generation from system models," 2011 IEEE Aerospace Conference Proceedings, AERO 2011, May 13, 2011.

<sup>3</sup>Carrion, C., Delp, C.L., Illsley, J., and Liepack, O., "Use of Operational Scenarios in Architecting MOS 2.0", SpaceOps 2010 Conference Proceedings, Huntsville, AL, Apr 25-30, 2010.

<sup>4</sup>Bindschadler, D., Delp, C., McCullar, M., "Principles to Products: Toward Realizing MOS 2.0", AIAA Space Operations Conference Proceedings, June 2012.

<sup>5</sup>Delp, C. L., Bindschadler, D., Wollaeger, R., Carrion, C., McCullar, M., Jackson, M., Sarrel, M., Anderson, L., Lam, D., "MOS 2.0 - Modeling the next revolutionary mission operations system," 2011 IEEE Aerospace Conference Proceedings, AERO 2011, May 13, 2011.

<sup>6</sup>Khan, Dubos, Tirona, Standley, Model-Based Verification and Validation of the SMAP Uplink Processes, IEEE Aerospace Conference Proceedings, March, 2013.