Systems Engineering Interfaces: A Model Based Approach

Elyse Fosse, Christopher L. Delp
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
elyse.fosse@jpl.nasa.gov

***Abstract*—The engineering of interfaces is a critical function of the discipline of Systems Engineering. Included in interface engineering are instances of interaction. Interfaces provide the specifications of the relevant properties of a system or component that can be connected to other systems or components while instances of interaction are identified in order to specify the actual integration to other systems or components. Current Systems Engineering practices rely on a variety of documents and diagrams to describe interface specifications and instances of interaction. The SysML[1] specification provides a precise model based representation for interfaces and interface instance integration. This paper will describe interface engineering as implemented by the Operations Revitalization Task using SysML, starting with a generic case and culminating with a focus on a Flight System to Ground Interaction. The reusability of the interface engineering approach presented as well as its extensibility to more complex interfaces and interactions will be shown. Model-derived tables will support the case studies shown and are examples of model-based documentation products.**

TABLE OF CONTENTS

1. INTRODUCTION

The topic of Interfaces is at the heart of the multi-disciplinary nature of Systems Engineering. This area covers what is necessary in order to connect the individual pieces of the System together into a working System. To accomplish the connection the relevant properties and behavior of each part of the System must be specified. It is also necessary to specify the particular connections between each part and the nature of those connections in terms of limitations, protocols, and various operational conditions or scenarios.

The describing and specifying of interfaces in a general way is a challenging problem. Current Systems Engineering practices rely on a variety of documents and diagrams to describe interface specifications for different Systems as well as existing industry standards for specific kinds of interfaces. SysML[1] provides a precise model based representation for specifying the interfaces of parts and integration between parts through those interfaces. The language also allows for domain-specific semantics to be applied as an extension to the basic modeling capability of SysML.

The interface engineering work performed by the Operations Revitalization (Ops Rev) Task[2], sponsored by the Multimission Ground Systems and Service Office of NASA, utilized a model based approach to represent the interfaces and interactions required for a Mission Operations System (MOS). Mission Operations Systems Engineering concerns with interfaces begin with the Flight-Ground Interface. The interface focuses on the interaction between the Flight System and the Ground System. Within the Ground System the MOS must interface with ground stations, networks, and a host of organizations. Within the MOS, Mission Services, Operations Roles, and Software must interface with each other.

This paper will describe the key Viewpoints used to define the patterns for modeling MOS interfaces. These Viewpoints will then be complemented by examples from the MOS 2.0 Architecture [2]. Beginning with the Flight-Ground Interface, examples from these models will be used to illustrate specification of Interfaces, Interactions between Interfaces, and the limitations and operational contexts of those interactions.

2. INTERFACE PATTERNS AND POINTS OF VIEW

Ops Revitalization is building a model of a Mission Operations System (MOS). The interfaces for Mission Operations Systems cover a broad range of systems, software, hardware, and human interactions. In order to model this broad range of interfaces and interactions, it is useful to describe the system from different points of view [3]. Ops Revitalization has developed the Mission Service Architecture Framework (MSAF) [4]. The MSAF defines the atomic interface engineering components (Table 1) as well as a set of patterns for modeling this broad set of integrated systems and other patterns in the MOS. These basic Views can be used to describe the different interfaces related to the MOS.

Table 2 identifies the Viewpoints that address important concerns with respect to the topic of interfaces. Part Interface Viewpoints of the system describe what interfaces the part presents. The Interface Layered Viewpoints describe the interfaces in a recursive manner and are a special case of the Part Interface Viewpoint. In systems where software is part of the system, it is common to have logical layering of software interfaces on top of hardware. Interface Specification Viewpoints describe the individual specifications of

978-1-4577-0557-1/12/$26.00 ©2012 IEEE.

[1] IEEEAC Paper #2562, Version 2, Updated 5/1/2013.

Table 1. Interface Engineering Definitions

Term	Definition
Interface	The system boundary that is presented by a system for interaction with other systems.
Interface Specification	Describes the nature of the boundary presented by a system or component in terms of properties and functionality.
Interaction	An instance of an operational entity (system, organization, or services) interface. The interaction can connect to other operational entities according to its Interaction Specification.
Interaction Specification	Describes how an operational entity (system, organization, or service) can effect another operational entity when a connection exists.

Table 2. Viewpoints

Viewpoint	Purpose	Concern
Part Interface	Identify Interfaces for a given Part.	What are the interfaces for a given Part?
Layered Part Interface	Specify layering of interfaces such as application, protocol and data layers.	What is the structure of the different information aspects on the interface?
Interface Specification	Specify a given Interface in terms of functions and properties of that interface.	What is the detailed set of functions and properties of a given interface?
Interface Connection	Specify the integration of 2 or more Parts through their respective Interfaces in terms of the specific conditions and function occurrences that define the integration according to the Interface Specification.	What Parts are connected to each other?
Interface Object Flow	Specify how objects (materials, information) flow across a given integration of interfaces for a set of Parts.	What are the flows between parts of the system?
Interface Function Occurrence	Specification for behavioral interaction across interfaces.	How do functions occur between Parts of the System?
Performance and Limitations on Interfaces	Specify constraints on interfaces such as policies, agreements and performance constraints.	What are the expectations and limits of the given integration?

each Interface. The Part Interface, Layered Part Interface, and Interface Specification Viewpoints describe the part of the system in terms of interfaces without explaining how they will be integrated. Providing these views of the system allows the Systems Engineer to evaluate and analyze parts based on their interfaces separately from how they will be integrated.

The remaining set of Viewpoints in Table 2 refer to how Interfaces may be instantiated into Interactions in order to describe context-specific integrations and behavior. Interface Connection Viewpoints identify what parts will be integrated with each other and how they will behave when they are integrated. Interface Object Flow Viewpoints describe how objects are allowed to flow across the interfaces of the integrated parts in the context with which the parts are integrated in. The objects could be physical objects, like shipping a spacecraft to Cape Canaveral, or Radio Waves in deep space. The objects could also be logical like data moving through cables. Interface Function Occurrence Viewpoints show behavior interaction between parts. From this vantage point the system can be described in terms of the occurrence of functions and events among the interfaces of components. Interface Constraints Viewpoints describe how the interface is constrained. This could be in the form of expected performance or operational limitations. Together, these Viewpoints describe integration of parts through their interfaces.

3. MODELING INTERFACES WITH SYSML

The SysML specification provides a precise model-based representation for interfaces and interactions. Interface specifications are described using SysML flow ports, operations, and signals which correlate to the system or component interface properties, as shown in Table 3.

Table 3. Interface Specifications To SysML

Interface Property	SysML Property
Functions	Operations
Signals	Signals
Information I/O	Flow Ports
Materials I/O	Flow Ports

A simple system will be used to describe the Ops Rev Tasks's SysML interface engineering implementation. The following figures and tables are views into the SysML model describing the system shown in Figure 1, which is comprised of two systems, A and B. The previously identified viewpoints will be elaborated presently using System A's interface and its interaction with System B

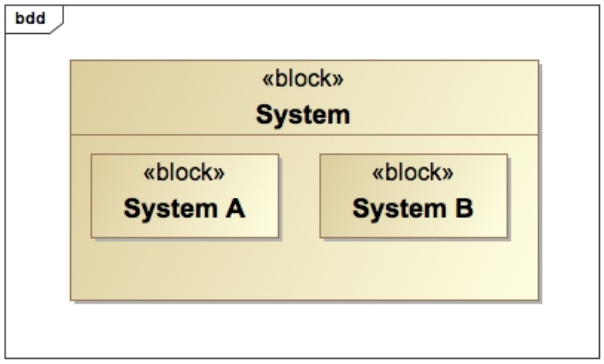

Figure 1. System Identification

Interface Specification Viewpoint

Every system or component that is intended to be composed into a greater system has some functionality that the comprising system needs. The functionality is a property of the system or component, independent from whom the system or component is intended to interact with. The Interface Specification Viewpoint focuses on identifying the functionality properties as operations owned by a SysML block. The functionality identified is the generic set of functionality that is available to any system or component when an interaction exists.

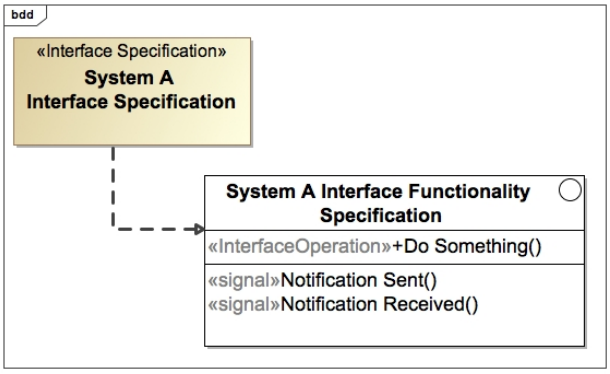

Figure 2. Functionality Specification

Figure 2 illustrates that an interface element named "System A Interface Functionality Specification" is used to describe the functionality of System A. The functions that are available for use by other systems or components during an interaction are shown on the interface element as operations. The parameters of the operation are suppressed in Figure 2 for the sake of readability and are instead shown in Table 4. Notifications are also specified on the interface element and are accessible during an interaction with System A.

The "System A Interface Functionality Specification" is realized by the "System A Interface Specification" which means that the functionality identified is available for use when a SysML port is typed with the "System A Interface Specification" block. Describing the functionality in this way allows for the reusability of the interface specification. Any SysML port that is typed by the "System A Interface Specification" will automatically have the functionality specified.

Note that Ops Rev utilizes the extensibility of SysML to create a Domain Specific Language (DSL) by extending SysML Block to "Interface Specification" and SysML operation to "Interface Operation", as seen as stereotypes (enclosed by guillemets) for their respective elements in Figure 2. Extending SysML and creating the appropriate customization classes allows for any properties or relationships that are true for any interface specification or interface operation to be created once and be reused by applying the appropriate stereotype. For example, if all interface specifications have some property that identifies the time for which the specification is valid, then this property can be a part of the block that is associated with the customization class for the "Interface Specification" stereotype. When the stereotype is applied to another element, the element will already have the time property associated with it.

Extending SysML in this manner also allows Systems Engineers implementing the interface pattern to now work and implement SysML elements that make sense for their domain, such as "Interface Specification" or "Interface Operation".

Table 4. System A Operation I/O

Operation	Inputs	Outputs
Do Something	Raw Input	Manipulated Output

Table 5 identifies the type of information parameters used in System A interface operations. The parameter types are themselves SysML Blocks and can have properties specific to them such that when applied as the type of a parameter that parameter has those same properties. The overall pattern that is emerging is that SysML provides the ability to specify properties in a recursive fashion (an interface has some information property which in turn has some other defining property, and so on). A Systems Engineer is able to identify and define properties at any level of fidelity that is necessary. For example, currently the "Information Specification" block that types the functional inputs and outputs has no properties explicitly identified, which is assumed to be the correct level of fidelity. If at a later point in time a Systems Engineer identifies some property that is true for all "Information Specifications" the property can be added to the element in a single place in the model and all other elements that are typed by the "Information Specification" will receive that update.

Table 5. Operation I/O Information Types

Function Input/Output	Type
Raw Input	Information Specification
Manipulated Output	Information Specification

The model-derived figures and tables described in this section comprise a interface specification view for System A. This view could be incorporated into supporting interface documents or presentations by a Systems Engineer to provide an accurate and explicit definition of the interface functionality of System A.

Layered Interface Viewpoint

SysML flow ports on an interface specification prescribe some sort of logical layering which the Systems Engineer

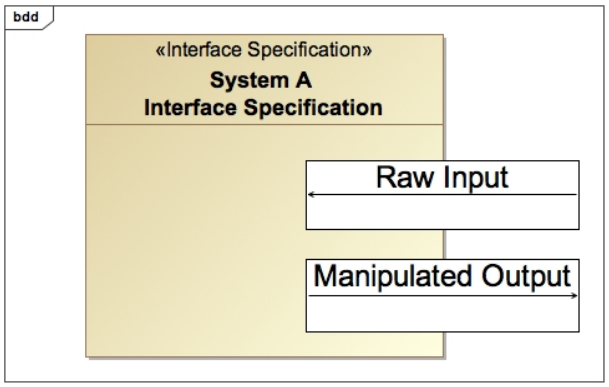

Figure 3. Information Interface Layer

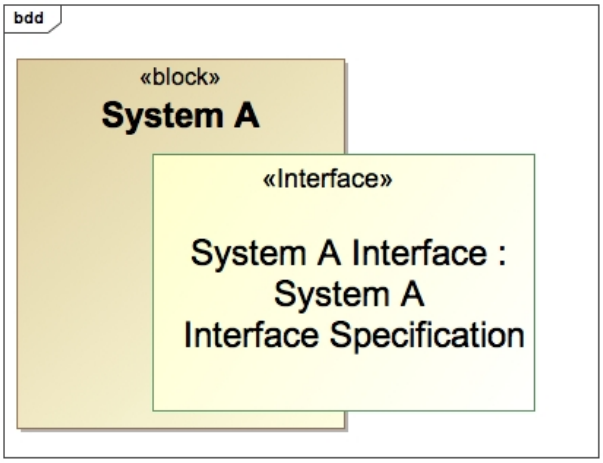

Figure 4. System A Interface

wants to abide by. Layering interfaces contributes to the separation of concerns methodology driven out by Viewpoints. In particular it is possible to create ports on an interface specification, thus creating nested ports. The ports nested upon the interface specification concern themselves with properties that provide greater detail about the specification.

Figure 3 illustrates how the "System A Interface Specification" further derives properties about the types of information System A expects or provides in order to fulfill its functionality. It follows then that the information identified as ports are related to the functional parameters identified (refer back to Table 5). This relation at the simplest level could be an equivalent, but could also be a subset relationship where the information inputs and outputs for the functions could be a subset of the information actually expected or provided on the port. The implication then is that there is some system functionality that is not used by another system or component, but is needed for that system to meet its objectives.

Interface Viewpoint

The properties designated by an interface specification are applied to a system's interface in SysML by assigning the type of the interface port to be that of the interface specification element. This means that the system specification has the details of the interface, regardless of with whom the system is expected to interact. Decoupling the system's interaction(s) and interface in this way allows a systems engineer to describe the concerns of the interface without convolving those concerns with that of interactions. The implication is that a system's interface can be identified and specified once with as many instantiations as needed to achieve the expected interactions.

Figure 4 shows that the System A port stereotyped "Interface" is typed by the "System A Interface Specification" that was previously defined. The assignment means that "System A Interface" has all of the functionality and information properties as defined by the "System A Interface Specification." The interface for System A has been fully defined with no regard to System A's interactions. As interactions with System A are identified, the "System A Interface" is instantiated and refined to meet the explicit needs of the specific interaction.

Interface Connection Viewpoint

The method of instancing an interface involves redefining the interface specification into an interaction specification. The redefinition involves redefining port and operation properties to meet the specific needs of the interaction. The assertion is that all instances contain a subset of the properties defined in the interface specification. Mathematically, let $\mathbf{S}$ be the set of interface properties specified by an interface specification and let $\mathbf{S}^{'}$ be the set of interaction properties specified by an interaction specification. Based on SysML redefinition rules, it follows then that, $\mathbf{S}^{'} \subseteq \mathbf{S}$.

Figure 5 shows that the System A interface specification is specialized to be an interaction specification for the exchange between System A and System B. For completeness all properties of the interface need to be inherited as well and as a result the functionality of System A's interface is specialized to be specific for the exchange. "Functionality for System B" redefines the interface operation found on "System A Interface Functionality Specification". The redefinition means that the the properties of the "Functionality For System B" interface operation, such as its parameters, are either of the same type as the redefined interface operation "Do Something" or have a type that further refines the type that is being redefined. The information implemented in the interaction specification is a specialization of the block used to type information in the interface specification. Similarly, the information property "schema" refines the generic property "information property". Completing the redefinition involves creating a new port on the System A block and having that port redefine the "System A Interface" port. Additionally, the new port is typed with the newly instanced interaction specification. System A now has the port needed to fulfill its need to interact with System B.

Creating the interface and interaction with this model based approach allows a Systems Engineer the ability to implement automation (exploiting the patterns and stereotypes used) in order to completely and rapidly redefine an interface for each necessary interaction. The automation employs the DSL to capture and replicate (e.g., specialize) inheritance, composition, and dependency relationships and the affiliated model elements, thus ensuring that all interactions are implemented in a similar manner and thereby removing any ambiguity from their specifications or descriptions.

Figure 6 and Table 6 together describe the functionality occurrences in System A's interaction with System B. The figure shows explicitly that there is a connector between the

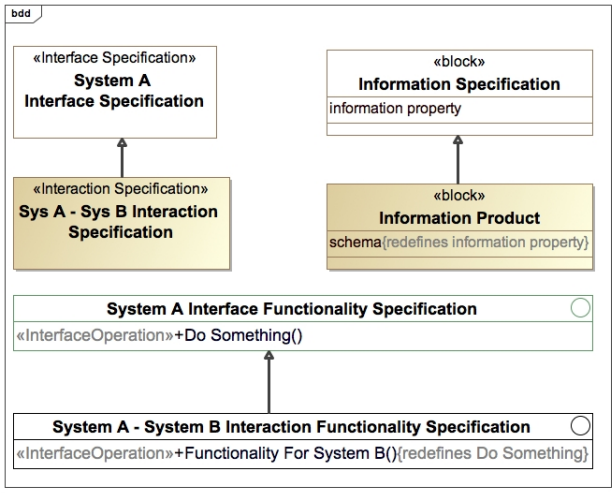

Figure 5. Complete Interface Instance

Table 6. Interaction Functionality Description

Interaction Spec	Inputs	Outputs
Sys A - Sys B Interaction Spec		
Functionality for System B	I.P. B	I.P. A
Sys B - Sys A Interaction Spec		
System B Interface Operation	I.P. A	I.P. B

two interface instances. The table implements a nested table technique such that the interface is identified and, indented within the next row, are the functional properties of the interface as well as the operation's corresponding inputs and outputs. For conciseness, "I.P." means "Information Product" and "Spec" means "Specification".

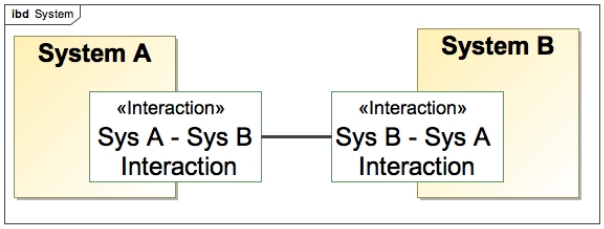

Figure 6. System Interaction

While System B's interface and interaction specifications weren't derived explicitly, the assumption has been made that System B has a single interface operation and its inputs and outputs are the exact conjugates of System A's inputs and outputs.

Interface Object Flow

Figure 7 and Table 7 together provide an information exchange view of System A's interaction with System B. Each interaction is expanded to show their respective nested flow ports, which identify the expected inputs and outputs that occur during an interaction between the two systems. All information described in Tables 6 and 7 have "Information Product" as their type, which is also shown in Figure 7 as the information that is conveyed during the interaction. Connecting the interactions and typing the connections with the information flows completes the interaction definition. It can be seen then that an interaction is comprised of two or more interface instances with conveyed information connections.

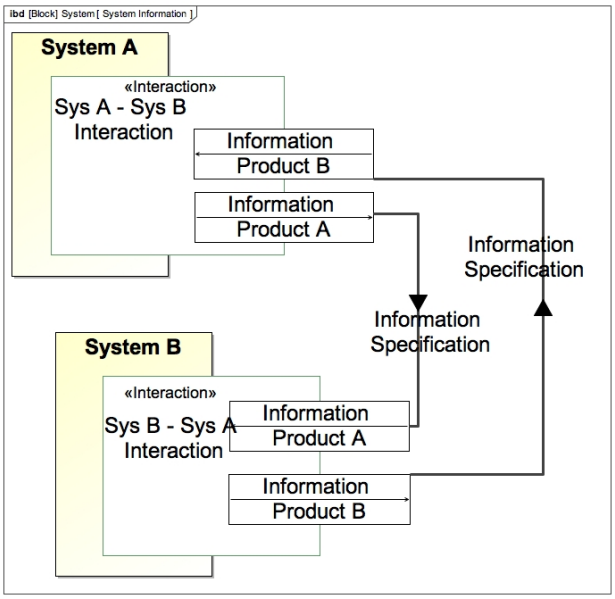

Figure 7. System Information Interaction

Table 7. Interaction Information Description

Interaction Spec	Inputs	Outputs
Sys A - Sys B Interaction Spec	I.P. B	I.P. A
Sys B - Sys A Interaction Spec	I.P. A	I.P. B

The instantiation process can be reused as many times as needed to specify all of System A's interactions. Thus this method allows the Systems Engineer to specify system or component interface properties once and redefine specific aspects as needed on an interaction-by-interaction basis. The redefinition part of the approach can be automated such that the Systems Engineer is able to focus on the aspects of interaction that are different and allow the automation to take care of the repetitive mechanics of interface instancing. The views shown can be queried from the model such that the Systems Engineer can provide stakeholders with accurate information about interfaces and interactions in a templateable manner.

Performance Limitations on Interfaces Viewpoint

Almost every interaction a Systems Engineer identifies has some sort of timing (e.g., duration or frequency) or quality (e.g., lossless) constraint that must be met in order for the interaction to be successful. To accommodate this, the model based approach is extended to include SysML constraint blocks as agreements among two or more systems or components. These agreements can contain all constraining parameters and are placed between the connectors of the interaction, as seen in Figure 8.

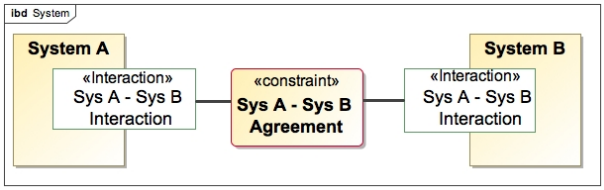

Figure 8. System Interaction with Agreement

The properties that pertain to the agreement could include but are not limited to the quality of the Information Products exchanged between System A and B as well as the timeliness with which System A expects to receive Information Products from System B.

4. Operations Revitalization Interface Engineering

The interface and interaction modeling approach was extended by the Ops Rev Task in order to represent interfaces and interactions required throughout mission operations as it pertains to a Multi Mission Operations System(MMOS). The necessary extension was for the pattern to be implemented in a deeper nested port manner (two levels of nested interfaces rather than one) in order to separate out control concerns that the Ops Rev Task is addressing. A Mission Operations Systems Engineer (MOSE) is particularly interested in how the MMOS interacts with other systems such as a Project or Flight System, as well as how the MMOS is organized internally based on interaction. The following section will show the Ops Rev Tasks implementation of the approach as it applies to a flight system and ground interaction, which is a high valued interaction for an MOSE to describe. Another implementation will be shown that focuses on how two Mission Services, components of the MMOS, interact with each other in order to achieve the MMOSs over-arching goal. The union of these implementations will show the generic approachs extensibility and reusability.

The description that follows is comprised of views, each of which conforms to a previously described viewpoint. The absence of a viewpoint can be understood as future or ongoing work for the Ops Rev Task.

MMOS - Flight System Interface Connection View

The interface specification for the MMOS is assumed to be fully specified and therefore ready for instancing. The assumptions made on the interface are that the functionality available is delegated to the MMOS from its component Mission Services (i.e., the MMOS does not carry out the functionality but delegates the appropriate work to the appropriate Mission Service), and that the functionality parameters as well as information required or provided are types of timelines[4]. The specification of the Flight System interface is outside of the purview of the Ops Rev Task and as a result the functionality is undefined. Figure 8 and Table 7 together provide a functional view of the MMOSs interaction with a Flight System. The functionality shown to be that of the MMOS is delegated to the MMOS by one of its component Mission Services, the Mission Engineering Service. For conciseness, Cfg means Configuration, Telem means Telemetry and portions of the MMOSs interaction functionality are hidden.

There is added complexity compared to the previous section due to the inclusion of the Ground Domain. The Ground Domain acts as a delegate for the MMOS interaction with the Flight System, and as such nests the MMOS Interaction with the Flight System onto its own interaction port with the Flight System. That is, the Ground Domains interaction with the Flight System specification includes a property that is the MMOSs interaction with the Flight System specification. For this reason, Table 8 excludes the Ground Domains interaction specification in order to eliminate redundancy.

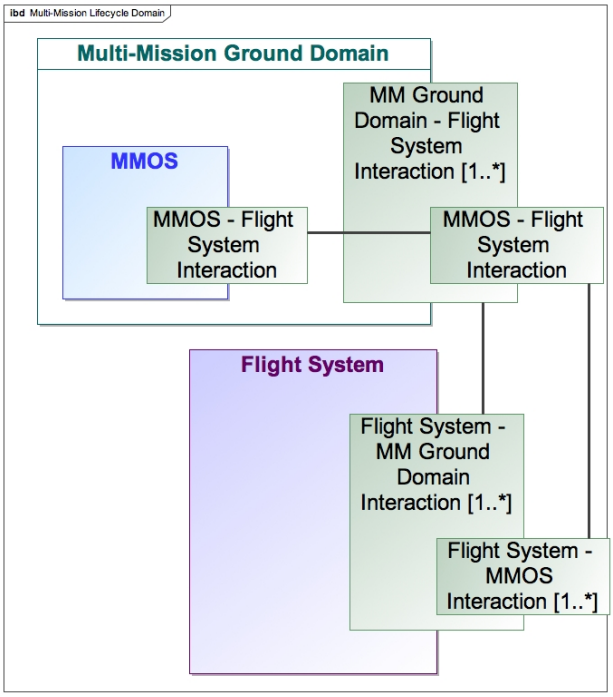

Figure 9. MMOS Functionality Interaction

Table 8. MMOS Interaction Functionality Description

Interaction Specification	Inputs	Outputs
MMOS - FS Interaction Spec		
Update Commands	Cmd	Telem
Update Configuration	Cfg Loads	Cfg Telem
FS - MMOS Interaction Spec		
N/A		

MMOS - Flight System Object Flow View

Figure 9 and Table 9 provide an information exchange view of the MMOS's interaction with a Flight System. There is added complexity compared to the previous section due to the addition of a "MMOS-Flight System Controller Interaction" port. The Ops Rev Task sought to not only separate interface and interaction functional and informational concerns, but also to further decompose information concerns as to the type of control trying to be achieved. A controller/controlled interaction is one type of control that the task has identified. Note that the conveyed information is an information type specific to Mission Operations such that the notion of Timeline[5]is extended to include mission operational concepts as conveyed information between the MMOS and other systems.

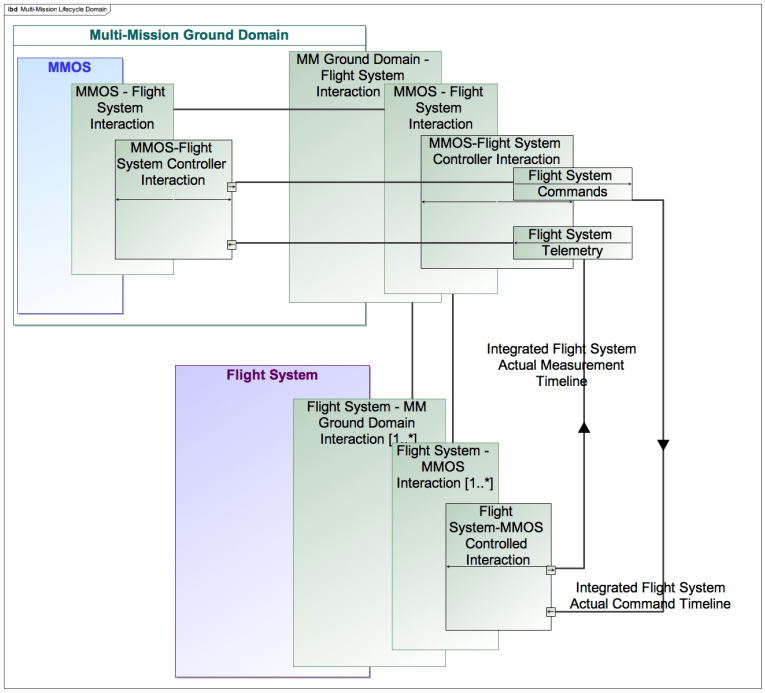

Figure 10. MMOS Control Information Interaction

Table 9. MMOS - FS Interaction Information Description

Interaction Spec	Inputs	Outputs
MMOS - FS Controller Int Spec	FS Telem	FS Cmds
FS - MMOS Controlled Int Spec	FS Cmds	FS Telem

MMOS Internal Interaction Object Flow View

The Ops Rev Task also implemented the same interface engineering approach to specify interfaces and implement interactions for MMOS components, known as Mission Services. Mission Services exert a different type of control on each other, called director/directed control. Figure 4 is a view of how the MMOS's Mission Engineering Service(MES) interacts with the MMOS's Flight Systems Engineering Service(FSES). The information identified in the view describes information pertaining to state, which is a subset of the information that may be conveyed between these two Mission Services in pursuit of directional control. Information pertaining to commands and activities would also need to be included in this view for it to be considered complete. For conciseness, the supporting tables are suppressed here but can be queried and rendered out of the model.

This section has shown that the generic interface engineering approach was competent enough to be implemented in the context of mission operations for the Ops Rev Task. Not only was the approach sufficient, it was shown to be extensible and reusable. The approach allows the Ops Rev Task to reduce the amount of effort focused on creating documents , and instead focus on specifying the interfaces and interactions in a model-based way so that the corresponding views can be derived in a formulaic manner. Further automation endeavors have allowed the Ops Rev Task to capitalize on this approach by reducing the effort needed for the instantiation of interfaces.

Verification

A model-driven interface engineering approach provides the Systems Engineer with some useful verification functionality. When dealing with complex systems it is arduous for a human to confidently verify that all information exchanges are appropriate and correct. Using blocks to type the flow ports on the interfaces, as well as to type the parameters of operations, allows the Systems Engineer to query the model to check that all interactions have allowable types connected. Rules can be created that identify the allowable port type connections. The model can then be queried and a list of invalid connections can be produced. Further, implementing a generalization hierarchy to the blocks used as information types, as seen in Figure 5, affords the Systems Engineer the ability to use generic types early in the interface engineering task. These types can be later refined to include more specific information types as that information becomes available, or as it is needed to describe the interfaces and interactions at a prescribed level of fidelity.

Interface Function Occurrence Extension

The interaction across interfaces can be specified through SysML Sequence Diagrams. A current limitation of SysML is that it is difficult to associate operations on a block with functions calls on a Sequence Diagram. This limitation disallows that ability to have the functionality of a system traced across interactions. It is possible, with some adaptations and extensions to SysML to extend the model based approach to include function occurrences among two or more components. These occurrence would be shown on a Sequence Diagram along with their associated timeliness and quality constraints.

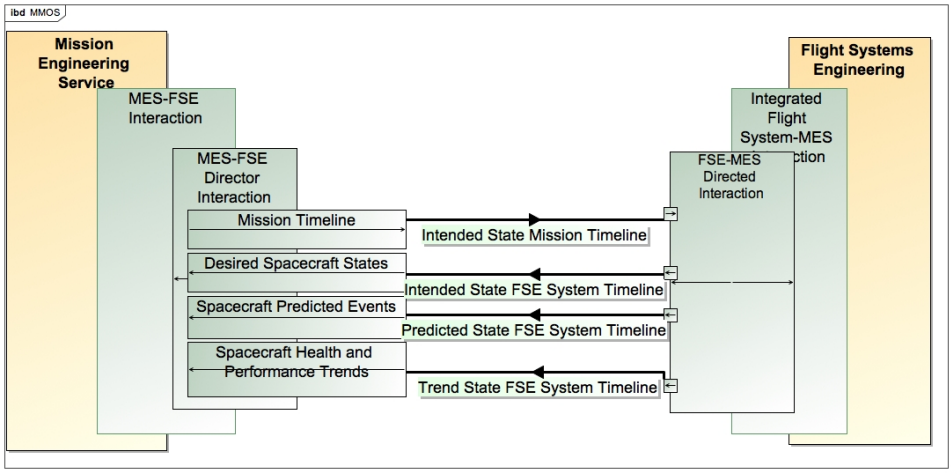

Figure 11. MES to FSES Interaction

5. SUMMARY

The Ops Rev Task implemented a model-based engineering approach to provide a more rigorous and formulaic description of interfaces and interactions. The task uses a separation-of-concerns methodology to both provide descriptions of the interfaces and interactions, as well as to decouple their specifications. The implementation was shown to be reusable in both a generic manner, as well as in its applicability to engineering the Multi Mission Operations Sys- tem(MMOS) interfaces for the Ops Rev Task. Its extensibility was also shown through the MMOS cases presented. The efficiency of the approach in terms of view generation and model verification has afforded the Ops Rev Task more time to focus on the engineering of the interfaces and interactions and less on document generation and narrative descriptions.

ACKNOWLEDGMENTS

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

REFERENCES

[1] Object Management Group, "OMG Systems Modeling Language (OMG SysML^{TM}), version 1.3," OMG, Tech. Rep. OMG document number formal/12-06-02, June 2012.

[2] D. Bindschadler, C. L. Delp, and M. McCullar, "Principles to Products: Toward Realizing MOS 2.0," in *Proceedings of the 12th International Conference on Space Operations*, Stockholm, Sweden, June 2012.

[3] ISO, "ISO/IEC 42010 Systems and software engineering-Architecture description," Tech. Rep., 2011.

[4] e. a. Chris L. Delp, "Mission Service Architecture Framework: Model Based Mission Operations Systems Engineering."

[5] S. H. Chung and D. Bindschadler, "Timeline-based Mission Operations Architecture: An Overview," in *Proceedings of the 12th International Conference on Space Operations*, Stockholm, Sweden, June 2012.

BIOGRAPHY

Christopher L. Delp *is the Systems Architect for the Ops Revitalization task in MGSS and a Lead Systems Engineer for MBSE on the Europa Mission. He is a founder of the Modeling Early Adopters grass roots Model Based Engineering working group. Chris continues to lead the INCOSE Space Systems Working Group MBSE Challenge Team for the INternational Council On Systems Engineering. Previously he served as Flight Software Test Engineer for MSL and Software Test Engineer for the Tracking, Telemetry, and Command End-to-End Data Services. He also leads the INCOSE Space Systems Working Group's entry in the Model Based Systems Engineering Grand Challenge. Additionally, he has performed research on software verification and tools for Service-Oriented Architecture in support of the Deep-space Information Services Architecture. Prior to coming to JPL, he worked as a software engineer performing DO-178b Level FAA flight qualified software development and testing on Joint Tactical Radio System (JTRS) and the T-55 Full Authority Digital Engine Controller (FADEC). Chris earned a Master of Science in Systems Engineering from the University of Arizona where he studied Model Based Systems Engineering, Simulation and Software Engineering. Previous to graduate studies, Chris performed his duties as a systems engineer on Missile Systems Verification and Validation.*

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