

Extension of MBSE for Project Programmatic Management on the Asteroid Redirect Robotic Mission

Oleg Sindiy, Brian Weatherspoon, Raffi Tikidjian and Tanaz Mozafari
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109

{Oleg.V.Sindiy, Brian.M.Weatherspoon, Raffi.P.Tikidjian, Tanaz.Mozafari}@jpl.nasa.gov

Abstract—Model-based Systems Engineering can be employed beyond management of the technical architecture development of a system to also manage the programmatic associated with Systems Engineering activities of a project. On NASA’s Asteroid Redirect Robotic Mission, MBSE has been successfully employed to manage, generate, and interact with the documentation-based deliverables associated with System Engineering activities. This has been involved in defining and tracking project document, milestone, and personnel metadata via the same modeling framework used for the technical architecture management. Additionally, it has focused on improving overall user experiences through linkage of documentation to technical content in the system model, automation of manually intensive tasks, and others stakeholder-oriented features.

TABLE OF CONTENTS

1. INTRODUCTION	1
2. MODELING FRAMEWORK	2
3. DOCUMENT MANAGEMENT	3
4. DOCUMENT METADATA DEFINITION	5
5. PERSONNEL METADATA DEFINITION	8
6. CONCLUSIONS	9
7. FUTURE WORK	9
ACKNOWLEDGEMENTS	10
REFERENCES	10
BIOGRAPHIES	10

1. INTRODUCTION

While most of the existing literature addresses Model-Based Systems Engineering (MBSE) in the context of technical architecture development, MBSE can also play a significant role in managing the programmatic associated with Systems Engineering (SE) activities of a project. That is, MBSE tools and products can also facilitate automation and improvements to the overall user and stakeholder experiences within a project life cycle. These same tools can also provide a means to manage, generate, and interact with the documentation associated with any SE undertaking. As such, while the application of MBSE for the technical architecture development of the proposed Asteroid Redirect Robotic Mission (ARRM) has already been described in earlier (in Ref. [1]) this paper will describe the related use of MBSE in programmatic, user and stakeholder experiences, automation, and project documentation generation. Specifically, this paper will describe how the ARRM team

has leveraged MBSE capabilities to manage the project programmatic such as:

- document metadata definition for use in document cover page generation and for tracking document ownership, approval, and release state information,
- project metadata definition for use of personnel role descriptions and assignments, and
- status of, including release schedule reporting with regards to project milestones and of course, access to latest in-work and latest approved project documentation.

Overview of the Asteroid Redirect Robotic Mission

The proposed Asteroid Redirect Robotic Mission calls for capture of an asteroid boulder and its redirection to an astronaut-accessible orbit around Earth’s moon [2,3]. Launching in the early 2020s, the ARRM’s Asteroid Redirect Vehicle (ARV) would cruise to a large Near Earth Asteroid (NEA) using a state-of-the-art Solar Electric Propulsion system. The ARV would characterize the asteroid with on-board instruments and provide data to support the project asteroid team’s selection of the boulder from several candidates. The ARV would then land on the asteroid, secure, and retrieve a multi-ton boulder with the arms on the ARRM Capture Module (as illustrated in Figure 1), and ascend to an orbit around the NEA.



Figure 1. Artist concept for Asteroid Redirect Robotic Mission operations on an asteroid. Courtesy of NASA.

With the increased mass of the captured boulder, the ARV would perform a gravity tractor asteroid deflection maneuver. The purpose of this demonstration is to check the feasibility of employing this type of maneuver to protect Earth from potential future asteroid impact. The ARV would then return, with the boulder, to a crew-accessible orbit in cis-lunar space. An astronaut crew, of the Asteroid Redirect Crewed Mission (ARCM), would be launched in the Orion capsule in mid-2020s to rendezvous with the ARV and captured boulder. As illustrated in Figure 2, the ARCM crew would perform extravehicular activities to study the asteroid material and collect samples for return to Earth for further investigation.

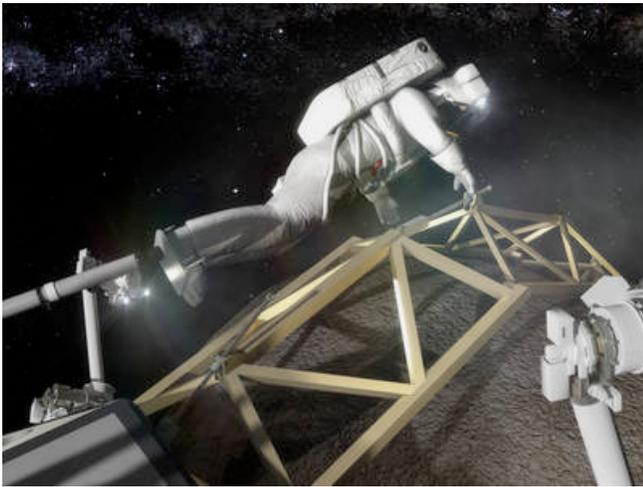


Figure 2. Artist concept for ARRM operations with an Asteroid Redirect Crewed Mission crew and captured asteroid boulder. *Courtesy of NASA.*

In accomplishing its mission objectives, the ARRM would demonstrate the use of new technologies, advance the application of existing technologies, and showcase the advantages of joint robotic and human space exploration programs.

Summary of Previous Work

MBSE seeks to provide a *single-source-of-truth* approach, where the content managed in a centralized System Model, and its derived products, becomes the de-facto source of project information. A System Model can describe constituent components, relationships, interfaces, and/or ownership/responsibilities—that is, the technical architecture. The use of MBSE for technical architecture development on ARRM can be found in Ref. [1], where the paper provided an overview and examples of the targeted MBSE deployment for development of the mission operational concept, system description, and functional requirements. The core types of SysML elements and relationship addressed by the technical architecture development in the ARRM System Model are shown in Figure 3.

However, a System Model can also be leveraged to capture information on the team organization and responsibilities, project risks, schedules, and/or budgets—that is, the programmatic information of a project. Thus, a well-

maintained and content-rich System Model can capture and manage both the technical and the programmatic rationale for the system design.

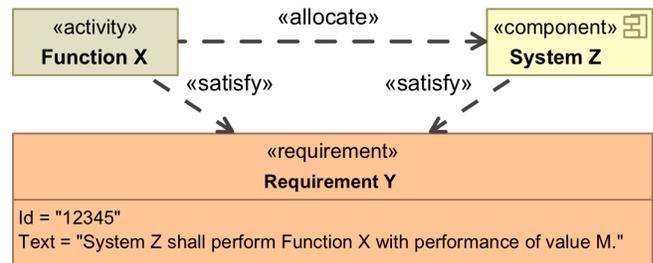


Figure 3. Core elements and relationships within the technical architecture of the ARRM System Model.

Reference [1] introduced the use of a versatile product reporting infrastructure with both web-based (e.g., View Editor [4]) and delivery of traditional products (e.g., reports in pdf, lists in tabular files) extracted from a System Model. This paper will further elaborate on the use of these for ARRM project programmatic.

Paper Organization

The first section in body of this paper summarizes the Modeling Framework employed on ARRM in support of technical and programmatic needs of the project. Next, Section 3 describes the applied Document Management process and in compliment, Section 4 describes the supporting use of Document Metadata Definition, including the related Project Schedule metadata. Similarly, Section 5 discusses the use of Personnel Metadata Definition for supporting programmatic administration via MBSE. Last but not least, Conclusions and Future Work are provided in Section 6 and Section 7 respectively.

2. MODELING FRAMEWORK

The selected modeling framework employs System Modeling Language (SysML) [5] to describe the technical and programmatic content captured in a ARRM System Model. The modeling framework employs characterizations to define qualitative and quantitative properties and attributes for identified classes of elements. Specific instances of programmatic characterizations, and use of selected modeling patterns, will be described in subsequent sections.

Usage of Inheritance Techniques

The ARRM System Model utilizes a system of *inheritance* (i.e., defining a general class within the model and creating specialized versions of that class that inherit the properties of the general class) to maintain conformity throughout the modelling environment. This strategy introduces a level of robustness into the model, as it allows specialized object to maintain the same properties as their generalized object, as well as allows for rapid creation of new objects or patterns of objects that are intended to extend an already-existing object. An example use of inheritance in the ARRM System Model is shown in Figure 4. Here, a general

project:ExternalDocument class is specialized by a *Project:NASADocument* and *project:JPLDocument* classes, with Document ID and Revision properties, which are then further instantiated into applicable documents to the ARRM project. Inheritance has been used similarly in the management of *project personnel*, *project roles*, *project documents*, and so on. Given that the MBSE environment for ARRM is a collaborative one, strategic use of inheritance techniques has been key in maintaining a robust, uniform model over time. Moreover, this additional uniformity and robustness in the System Model database has been key in the automation of many modeling tasks; this is discussed next.

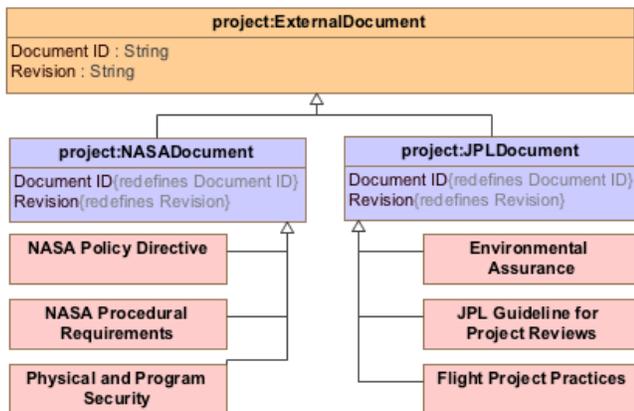


Figure 4. This inheritance pattern is used in the ARRM System Model in the modeling of reference and applicable documents that are referenced in project deliverables.

Automation of Daily Work

MagicDraw (MD) [6], the ARRM project’s selected modeling tool, contains a plugin called *MacroEngine*. This plugin allows a model-based systems engineer to build and employ automation scripts in areas of their work. The ARRM modeling team has employed this functionality for various daily tasks—especially those that previously required manual, time-intensive labor. These tasks include project personnel metadata synchronization (discussed in more detail later), requirements data management, data migrations between tools, and model/element maintenance. These tools have drastically improved the overall MBSE user experience, as many of the previously tedious or seemingly insurmountable tasks that served as a deterrent to adopting MBSE have essentially been automated. For example, in the early stages of ARRM, simply changing the string value stored within a specific property had to be done manually. While this is trivial in terms of time and effort for a single property, occasions arose when hundreds of properties needed to be updated with the same value at the same time (for example, an entire section of requirements might change “Owner” from one role to another; a value that is stored as a property of the individual requirement). This tedious task is one that has since been automated—a user can now use one of scripts to select all the properties for which they wish to change the string value, type in the new string value, and the tool will implement all of the changes in seconds (as opposed to minutes or hours). The same functionality has been extended to many similar use cases, ranging from element

and property name changes to entire modeling pattern restructures and refactoring. This automation capability has allowed for malleability within the System Model that can be leveraged when large, model-wide architectural changes are (inevitably, as project needs evolve) imminent and await implementation. Moreover, these tools incorporate Graphical User Interfaces (GUIs), which increase the overall usability of the tools and allows for use in a more generalized sense (i.e., a tool might be able to import *any* kind of model-data given that a user is able to enter some form of input through the GUI vs. specific types of data only assuming no user input is provided).

Expansion of Capabilities via Automated Data Migration

As mentioned earlier, scripting capabilities have assisted in facilitating data migration in between tools. Specifically, scripting has been used to facilitate data migration from non-MBSE, spreadsheet formats (such as Microsoft Excel or tabular comma-separated value, csv, files) into the ARRM MBSE environment. Existing functionality, while somewhat present in the MagicDraw tool, did not satisfy all the needs of the project to allow systems engineers to validly work in their tools of choice and still maintain an up-to-date, official MBSE single-source-of-truth database. Specifically, the MD capability allows a user to utilize spreadsheet data to create and import new elements, but it does not have the functionality present that allows a user to import *changes* to existing elements (e.g., changing the value of a property without creating a new, additional property to contain the new value). Therefore, as a result of a high learning curve that is present with the MagicDraw tool, many stakeholders performed the majority of their work in disjoint spreadsheets and other forms of Document-Based Systems Engineering (DBSE). The systems engineers (or their designees) were initially required to import their work into the MBSE environment manually; a tedious task. To solve this issue, a script was created that would facilitate the autonomous import of content from Excel, or any type csv file, into the ARRM System Model. While this tool was originally created to facilitate a mass transition of spreadsheet-based workflows to MBSE-based workflows, it transformed into a tool that is used on a daily basis to allow the capabilities of external, non-MBSE tools to be leveraged in ways that benefit overall productivity. The overall process is illustrated in Figure 5.

3. DOCUMENT MANAGEMENT

On ARRM, the majority of project documentation is defined and managed in the same MBSE-based environment as the technical content. The documentation process employs the DocGen plugin [7] in the web-based View Editor environment. Document titles, sections, tables, diagrams, and so forth, are simply just another set of elements in a larger System Model that can be used and managed in parallel with the technical content of ARRM. Thus, technical and programmatic content can be integrated into documentation, and then generated and tracked in real-time.

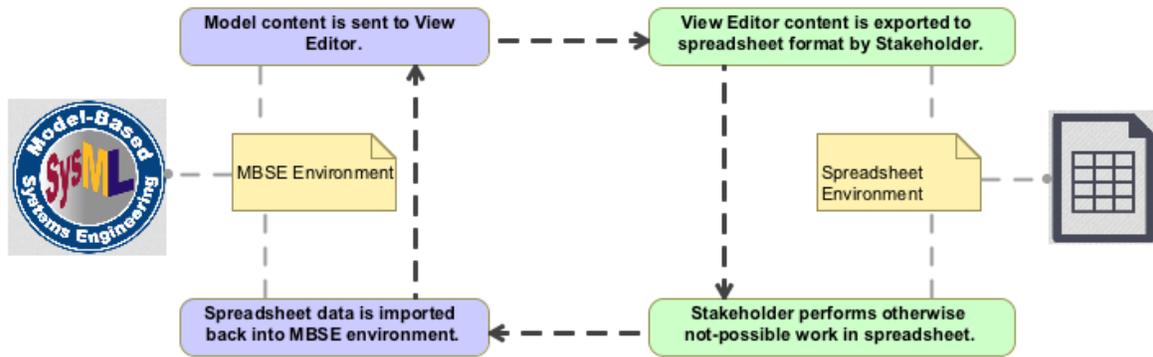


Figure 5. Workflow showing how Document Based Systems Engineering (DBSE) tooling advantages can be leveraged using automated tools created in the Model-Based Systems Engineering (MBSE) environment.

Project-Wide Content Reusability

Application of MBSE for project document definition and management has allowed for reusability and synchronization of document content. For example, through the use of the DocGen plugin in the web-based View Editor document management environment, document section reuse and content cross-referencing has allowed the project to keep common information up-to-date, and synchronized, across all ARRM project documentation. Using the MBSE approach of a *single-source-of-truth*, when actively defined content is updated by the content owner, the update propagates to all of the documents that use it. For example, content such as a high-level mission description can be defined in a single document <<view>> (i.e., a document section), and then that section <<view>> can then be added (i.e., reused) by many documents as needed. Similarly, the use of cross-referencing is paramount to keeping content synchronized and fresh across many documents that reference it. For example, document names and terminology definitions can be cross-referenced across various documents; when the values for that content is updated (in a single location), the cross-references automatically propagate (live update) to all of the effected in-work documentation in the View Editor environment. As such, a task that previously required extensive manual search-and-replace work, often throughout several disjoint document files and by many authors, is now seamlessly automated. Of note: since, in this environment, a single content change can propagate across multiple project deliverables, careful consideration should be given towards configuration control of content that is reused in multiple locations.

Versatility of Presentation Formats

In generating documents in a modeling environment, document authors are offered additional versatility in presenting the same technical content in various forms: tables, prose, diagrams, *etc.* For example, the same requirement title, shall statement, and rationale can be inserted (via cross-references) into a paragraph, table, or even a diagram (e.g., for parent-child requirement traceability flow). Similarly, numerical values for technical resources (e.g., mass, power, data) and programmatic data (e.g., cost, dates, authorship) can be called on and inserted into tables, paragraphs, or figures of choice.

Project-Wide Nomenclature Definition

Leveraging MBSE infrastructure and capabilities, a unified periodic collection and reporting mechanism for ARRM project-specific nomenclature was developed. The assembled project-wide Nomenclature list consists of acronyms, abbreviations, units, and glossary terms. While abbreviations, units, and glossary terms are populated selectively—via manual user review and inclusion—an automated script was developed to search for the large number of acronyms across all project documents maintained in the View Editor environment. The acronym search script periodically searches all project documentation. Once elements matching acronym criteria are found, the element is compared to existing list of acronyms. If not in the existing list, candidate acronyms are staged for user review for inclusion and definition. The resulting, assembled project-wide Nomenclature list is referenced by every document by insertion of a link to the independent document in View Editor, or the list is cross-referenced as a section in a document; the decision is left up to individual author's preference, as the ARRM project-wide Nomenclature list is quite extensive; it is in excess of 1000 elements and growing.

Modeled Document Linkages and Usages

Modeling the project SE deliverables within the ARRM System Model has allowed the project to create linkages between documents and other modeled elements, including requirements, personnel, and project schedule elements. While the full range of benefits, that these linkages provide, have not yet been leveraged (or even explored), these linkages have allowed for additional rigor in deliverables development that is generally not present in more traditional DBSE processes. For example, requirements developers can leverage the model infrastructure to cross-reference a separately modeled deliverable's name, where necessary. Hence, if this referenced deliverable's name is updated, there is no longer a need to correct the name of that deliverable wherever it was referenced; that is, this type of change automatically propagates.

4. DOCUMENT METADATA DEFINITION

Document Metadata

With regards to defining document stakeholders, the ARRM team created project-specific definitions for ARRM document Owner, Approver, Preparer, and Contributor roles that are emulated within the System Model. Within these definitions, a role owns or approves a document. Additionally, in this environment, the person filling a given role can change, over time. In contrast, a person can contribute to, or prepare, a document, regardless of what role they fill. In the ARRM System Model, class objects stereotyped as <<arm.Role>> are related to documents that signify owners or approvers, while “characterization” patterns are related to documents that signify document preparers and contributors (by embedding properties containing the person and their role). The characterization elements contain the personnel and role information of each preparer or contributor. Additionally, a given document is specified to have one owner (typically, the technical authority for the document), and one or more preparers. Furthermore, a document can have many approvers and contributors. The metadata pertaining to an ARRM document deliverable, as well as how that metadata is contained or connected, is presented in Figure 6.

Centralized Project Document List

For ARRM, MBSE is leveraged to maintain the associated project technical and programmatic deliverables (e.g., those pertaining to interfaces, requirements, and even project implementation and IT security). Metadata for these deliverables, such as the current state, release schedule, signatories, and identification numbers, are maintained within the System Model and displayed on View Editor in the form of an ARRM *Project Document List* (PDL). This list

acts as a central deliverables hub for the project, allowing ARRM stakeholders to view the latest documentation metadata, and even access the latest content (both, in-work or officially released), of any deliverable at any time. Furthermore, specific personnel (e.g., preparers and owners), depending on their roles, are able to interact with the PDL in order to make changes and/or add content. This functionality acts as a wrapper that allows non-MBSE experts to easily maintain this metadata while preserving its MBSE roots. Moreover, the use of MBSE in this programmatic area extends the *single-source-of-truth* infrastructure of the project, as this metadata is automatically updated any time it changes, and thus, is kept current with minimal maintenance beyond that of sustaining the source data.

Project Milestones, Phases, and Release Schedule

The project’s milestones and phases have also been captured within the ARRM System Mode. The captured metadata includes values for dates, names, and category metadata for milestones, and start and end date metadata for phases. Within the PDL, the milestone names and dates are used (via cross-references) to populate the fields pertaining to a deliverable’s release schedule (e.g., Draft Date, Preliminary Date, Baseline Date). Through these linkages, an integrated Receivables and Deliverables (Rec/Del) list is generated, in real time, in a centralized and easily accessible venue for the project stakeholders. As an example of how this is particularly useful for ARRM (and applicable to any other flight project), milestones are occasionally changed as a project lifecycle unfolds. In the MBSE environment, by simply updating the milestone element with a new date will result in an update to the PDL (and all relevant documents) with the new date. As such, this adds additional transparency to product delivery expectations when associated milestones that govern product release schedules change. Furthermore, much like with the Organizational Breakdown Structure, the

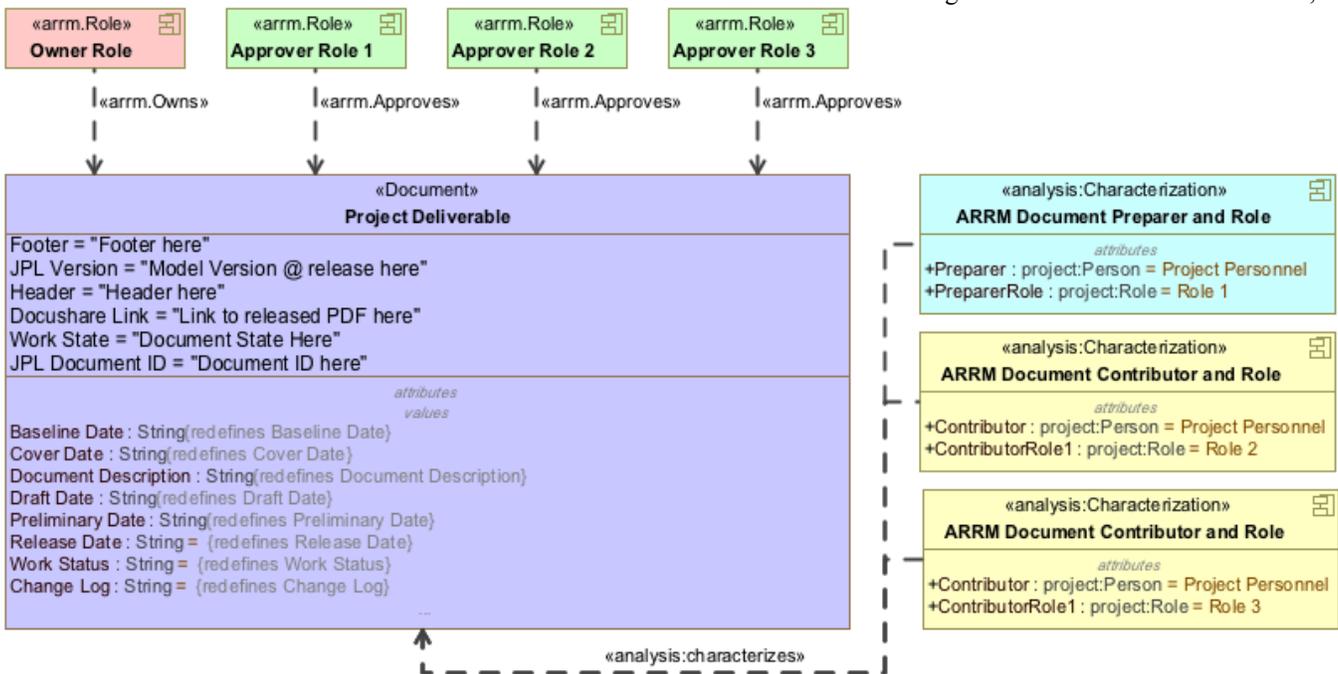


Figure 6. Diagram of document metadata with relationships used to relate respective roles or persons to the document.

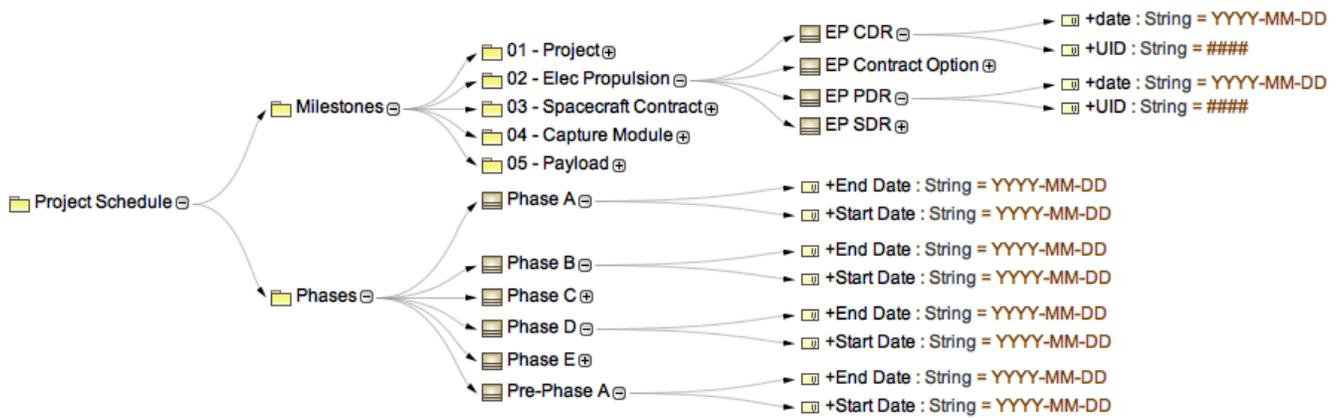


Figure 7. Example visualization of project phases and milestones with start and ending dates.

environment allows for visualizations of the data to aid with quality assurance and completeness. An example visualization of captured ARRM project milestones and phases is provided in Figure 7.

Autonomous, Uniform Formatting of Deliverables

The ARRM team has also leveraged MBSE to generate the front matter (e.g., cover page, signature page, table of contents, list of figures, list of tables, list of equations, and so on) of model-generated deliverables. Using the linkages from deliverables to roles and personnel mentioned previously, the front matter is automatically populated with the owner, preparer, approvers, and contributors, as well as any provided document metadata (e.g., change log, document ID, cover date, etc.). Additionally, signature lines are generated automatically for the owner and approvers. This format template is designed once in the model (using the DocGen plugin) and each document is then “plugged in” to this format template to generate its own specialized front matter based on its unique set of metadata. Any changes made to the core front matter format thereafter are reflected, in real time, in all the instantiated documents. In addition to this front matter, some common document formatting is also handled autonomously, including document headers and footers. Within headers and footers of ARRM deliverables, metadata (e.g., document ID, cover date, page numbers, and any marking language) is autonomously populated from model content when a pdf-version of a deliverable is generated.

Applicable and Reference Documents

As mentioned previously, ARRM utilizes a unique modeling pattern to capture non-ARRM external documents that are applicable to the project. These documents are modeled as simple Class elements that, as specified before, inherit attributes from a generalized *project:ExternalDocument* class. These attributes include the applicable document ID and revision used by the project (not necessarily the latest). Most importantly, when these documents are modeled for reference, the modeler retrieves, when possible, the hyperlink to the document that the element represents. When these external documents are linked to project deliverables within

the System Model (generally in a standardized “Reference and Applicable Documents” section), DocGen is leveraged to generate a table of these documents with hyperlinks embedded. This allows a reader to have direct access (when reviewing the document in a digital format, such as in View Editor or pdf) to those applicable documents. This aids in removing human error involved with searching for and locating the *correct document* referenced by the material, as often times multiple revisions of the same applicable document are readily available to a given reviewer at the same time. Note that no consideration is given to any access rights requirements; a user who is reviewing a document must have or obtain access to review the applicable documents.

Document Release Process

The ARRM document release management process, presented in Figure 8, is used to promote the project’s official documents from unreleased (in-work) to released (controlled) states. Prior to formal approval, release, and baselining, project documents are controlled by the individual responsible for their technical content (i.e., owners and preparers). In this process, a document’s author must request a document ID from a Configuration Management Engineer (CME) and submit a request for a blank document to be published to VE by the MBSE team. Then, when a document is ready for review, the latest document and modelled-content (where applicable) are published from the System Model to VE. Once the document has been reviewed and any requested changes dispositioned, it is ready for approvals. At this point, the MBSE Team creates a snapshot (also called a “tag”) from VE and creates a Portable Document File (pdf) from the snapshot, which the CME Team submits for official approval. Document approval and release is completed when required signatures are obtained using a customized workflow in the ARRM’s task management tool. The changing document states are maintained through the model system and displayed in View Editor in the PDL. In turn, this allows the project team to view the latest states of each document in the PDL, with direct links to the in-work and/or controlled versions of all of the listed documents.

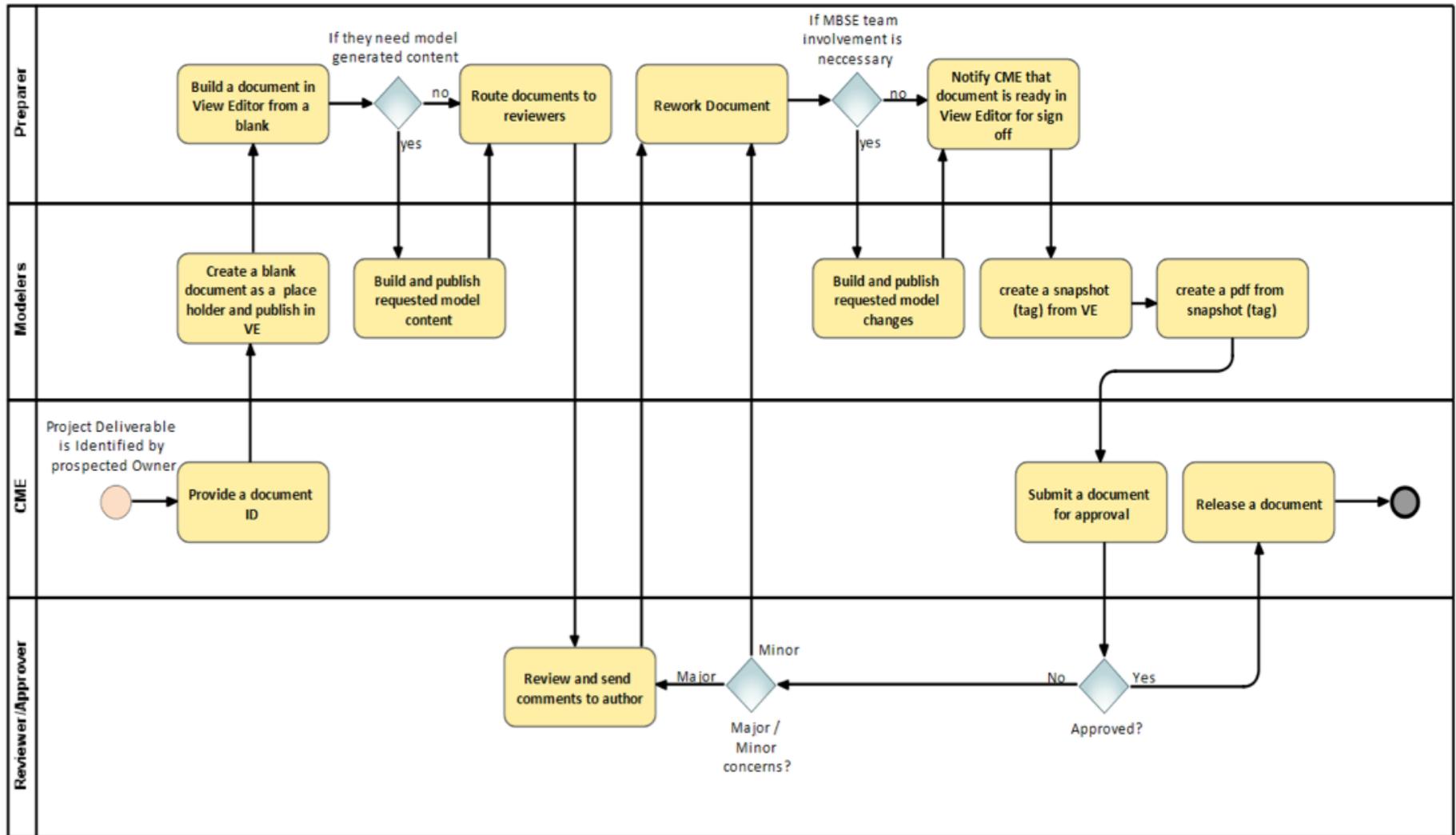


Figure 8. Document release process flow, from document instantiation to document release.

5. PERSONNEL METADATA DEFINITION

Over the course of modeling documents and other programmatic aspects of the ARRM project, the ARRM team found additional value in including project personnel, roles, and other pertinent personnel-related metadata in the System Model. Modeling project personnel has allowed the ARRM team to capture personnel-related information within the System Model, including relationships of personnel to documents, requirements, and other technical content, as well as maintain a current *Project Personnel List* (including contact information for all personnel) for ARRM project members to access and reference as needed. Specifically, the personnel list has provided a central directory for project personnel and contact information (regardless of their organizational affiliation or location), as well as aided in maintaining the project’s Lightweight Directory Access Protocol (LDAP) group membership. This list is viewable on View Editor by any project employee; and thus, receives consistent feedback pertaining to its accuracy. Using this approach, information pertinent to personnel or roles, such as which deliverables a specific person is responsible for, is captured and made accessible to project stakeholders in ways that are otherwise not possible on a geographically distributed, multi-center and multi-organization project such as ARRM.

Roles, Center Affiliations, and Role Assignments

As mentioned earlier, included in the metadata for project personnel is the center for which each personnel works and the role(s) each person has been assigned. On ARRM—a mission which spans multiple NASA centers and organizations—the inclusion of role assignments and center affiliations within the ARRM System Model has been particularly helpful in tracking which personnel are working on the project, as well as in which areas and what organizations and/or centers they work for. Furthermore, since role assignments often change, capturing the roles (and all associated information) within the System Model has allowed for greater ease in keeping role assignments up-to-date across the project (and displayed in the *Project Personnel List* mentioned previously). For example, when a role assignment changes from one personnel to another within the System Model, all deliverables referencing that role as an Owner or Approver automatically reflect who the role has been assigned to; a programmatic detail that previously was often out-of-date and required manual effort to maintain. While the periodic updating of the role

assignments is still performed manually, it only needs to be done one time within the System Model; rather than many times throughout all the various deliverables and supporting products associated with ARRM. The modeling of roles also allows for additional gap analysis, as personnel with multiple roles or roles with no assigned personnel are now clearly shown in diagrams and other presentation elements generated from the System Model. Furthermore, role descriptions for each role are captured in the System Model, which allows for greater transparency in the responsibilities inherited by an individual who assumes a given role on the project.

Organizational Breakdown Structure (OBS)

Roles contained in the ARRM System Model are organized in a manner so as to represent the Organizational Breakdown Structure (OBS) of the project. Through this organization strategy, the OBS can be shown through View Editor in a variety of formats (e.g., diagrams, tables, text) and thus, made available to the project stakeholders. A consolidated diagram example is shown in Figure 9. Note that this diagram is updated within the MBSE tooling environment autonomously once the diagram is initialized, and thus, it is consistently up-to-date and requires minimal effort to maintain. This approach has allowed the project to maintain a current, up-to-date OBS. To elaborate as to how the metadata is maintained, when a person is added to the project, they often (if not always) need access to various tools used on the project—this requires LDAP membership. To obtain membership, individuals must make a formal request to the project’s Configuration Management Engineer (CME), which must include their role(s) on the project and what tools they need access to. Upon approval, these individuals are added to the appropriate LDAP groups. The ARRM System Model is autonomously synchronized periodically with the ARRM LDAP groups (discussed later), and thus, the ARRM System Model reflects these changes in project membership. These changes are brought to the attention of the ARRM team, which provokes the team to define which role they fill; information that is captured in the original request. These roles are assigned, and all applicable diagrams and presentation elements reflect the change in content autonomously. This workflow is shown in Figure 10. Additionally, filters are employed to generate specialized role diagrams as desired by particular stakeholders.

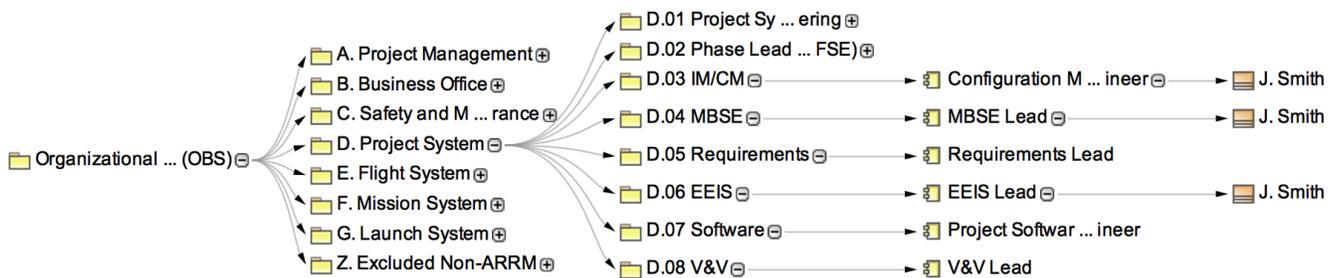


Figure 9. Example of an Organizational Breakdown Structure diagram generated from the ARRM System Model.

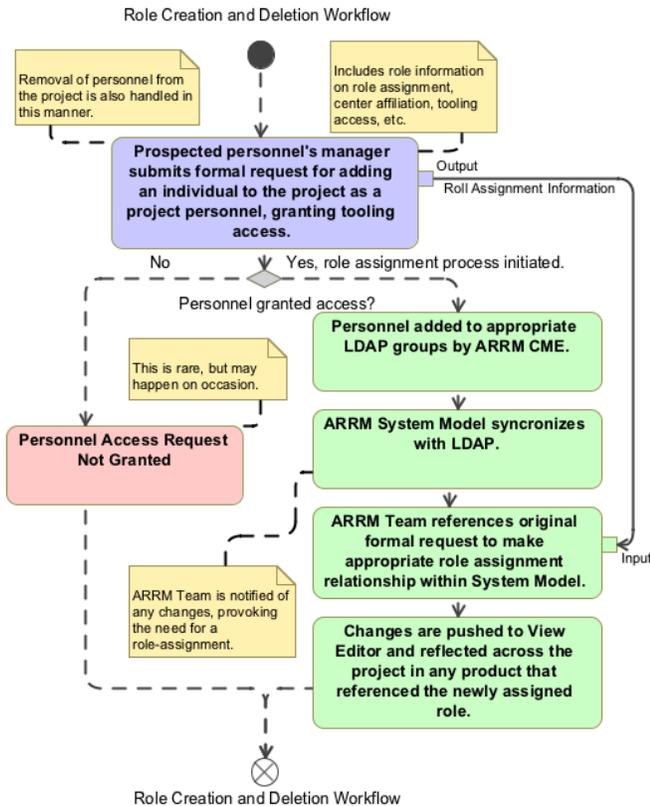


Figure 10. Process by which role assignment changes are disposed on ARRM with the use of MBSE.

Tool and Automation of Personnel Metadata via LDAP

Initially, ARRM project personnel and associated metadata was maintained manually within the model; frequently becoming out-of-date and very difficult to maintain. However, this type of data should be drawn from and controlled by an authoritative source (such as LDAP). By leveraging the tooling environment ARRM uses to perform MBSE, a script was created that automated and expanded the modeling of project personnel by synchronizing the model with the project's LDAP groups. These groups are actively maintained by the project via JPL's institutional infrastructure because they govern secure access to project resources. The employed tool creates and updates project personnel elements within the model by querying the LDAP server for user-specified group membership, as well as synchronizes useful metadata (shown in Figure 11) such as username, phone number(s), email, LDAP group membership, and center affiliation. This metadata, much like the PDL, is displayed on ARRM's View Editor website in a central place that all stakeholders can access. Furthermore, the information is cross-referenced all throughout the System Model, extending the *single-source-of-truth* infrastructure into the programmatic realm of personnel management. In this specific case, since metadata is imported from LDAP into the MBSE environment and never sent in the reverse direction, LDAP effectively functions as the official, single source of project personnel information rather than the ARRM System Model. In turn, this is an example of how MBSE can be employed in a multi-database environment.

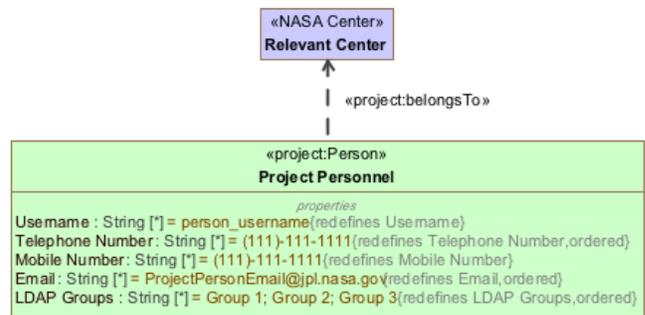


Figure 11. Metadata managed by the automated "LDAP Sync" tool, which pulls the data from the Lightweight Directory Access Protocol (LDAP) servers directly.

6. CONCLUSIONS

This paper described how MBSE can be expanded to manage the programmatics of the Systems Engineering activities of a project. Description of NASA's ongoing development of the Asteroid Redirect Robotic Mission was provided as an example of such an application effort for deliverables management, with modeling support, for document generation and management with associated metadata for the project's documents, schedule and milestones, and personnel.

7. FUTURE WORK

A few of the programmatic areas that the modeling team is seeking to further leverage MBSE for ARRM are as follows:

Automation of Nomenclature Term Extraction

As mentioned before, the ARRM's nomenclature is created within the System Model and displayed on View Editor in a central project nomenclature and glossary dictionary that can be referenced throughout the project. Currently, the ARRM team has automated the detection of new nomenclature throughout model content, as well as the creation of those terms within the System Model. However, the work involved with *cross-referencing* these terms where they are used throughout project deliverables has not yet been automated, and therefore currently requires a large time investment to maintain manually. Without the cross-references, much of the benefit of modeling the nomenclature objects is not yet leveraged, as any changes made to this content are not reflected everywhere that the relevant term was used. Therefore, the ARRM team's future work pertaining to MBSE in the area of nomenclature includes automating this cross-referencing process. This capability will allow systems engineers to spend less time populating changes to nomenclature definitions/acronyms, as changes to these cross-referenced terms will then be reflected anywhere they are cross-referenced autonomously.

Evolving Document Metadata Definition

In addition to the current metadata captured within the model, future work includes capturing *receivables*; i.e., capturing information pertaining to which deliverables are needed for

stakeholders to begin or continue work. For example, the ARRM Mission Manager is a receiver of the ARRM Mission Plan document. Capturing this information within the ARRM System Model would provide means to track “Receivers” similarly to how Owners, Approvers, Preparers, and Contributors are tracked in the document management process.

Receivables/Deliverables Schedule

As mentioned earlier, project deliverables are captured within the ARRM System Model in a centralized Project Document List. However, the ARRM System Model has not yet been extended to capture the whole list of project deliverables (particularly those that are delivered in later stages of project lifecycle), so there is ongoing work to model the remaining deliverables. Furthermore, the ARRM team aims to explore other forms of display formats for project deliverables, such as within a Receivables and Deliverables schedule organized by project phase and project milestone.

Autonomous Cross-Referencing

As mentioned previously, the actual cross-referencing of the Nomenclature and Glossary elements is currently a manual task and, consequently, has not yet been extensively carried out by many stakeholders. The ARRM MBSE team plans to automate this process within the MBSE environment, so as to eliminate the need to manually cross-reference content and update manually when content is copy-pasted View Editor-based documents. This automation capability will likely leverage a script written within MagicDraw using MacroEngine (similar to the aforementioned LDAP tool).

ACKNOWLEDGEMENTS

This task was managed out of the Jet Propulsion Laboratory, a division of the California Institute of Technology, under a contract with the National Aeronautics and Space Administration. ARRM is directed through the NASA Asteroid Robotic Mission (ARM) Program with funding from the Science Mission Directorate (SMD), the Space Technology Mission Directorate (STMD), and the Human Exploration and Operations Mission Directorate (HEOMD). ARM is part of the “Asteroid Initiative” that includes other funding from the NASA’s Office of the Chief Technologist. The ARRM project is led by JPL in collaboration with NASA’s Glenn Research Center, Goddard Space Flight Center, Langley Research Center, Johnson Space Center, Kennedy Space Center, and industry partners.

Work described in this paper leveraged MBSE infrastructure and processes developed at JPL through Integrated Model-Centric Engineering (IMCE), Systems & Software Computer Aided Engineering (SSCAE), Europa mission, and many other institutional processes modernization efforts. The authors thank members of the ARRM multi-center MBSE team: Benjamin Cichy of GSFC, Kathryn Trase, Vicki Crable, and Edith Parrott of GRC, and Charles Budney, Carl Steiner, Larissa Kupferschmidt, Chrisma Derewa, Farah

Alibay, Matthew Rozek, Robert Castillo, and Sanda Mandutianu of JPL.

REFERENCES

- [1] Sindiy, O.V., Mozafari, T., and Budney C.J., “Application of Model-Based Systems Engineering for the Development of the Asteroid Redirect Robotic Mission,” *AIAA Space 2016 Conference*, Long Beach, CA, 13-16 September 2016.
- [2] Gates, M., “Asteroid Redirect Mission Status,” NASA [presentation], URL: http://www.nasa.gov/sites/default/files/files/20150408-NAC-Gates-ARM-v8-1_TAGGED.pdf, 8 April 2015, [cited 18 August 2016].
- [3] Gates, M., Stich, S., McDonald, M., Muirhead, B., Mazanek, D., Abell, P., and Lopez, P., “The Asteroid Redirect Mission and Sustainable Human Exploration,” *Acta Astronautica*, Volume 111, June-July 2015, p. 29-36.
- [4] Delp, C., Lam, D., Fosse, E., and Lee, C.-Y., “Model Based Document and Report Generation for Systems Engineering,” *2013 IEEE Aerospace Conference*, Big Sky, MT, 2-9 March 2013.
- [5] “OMG Systems Modeling Language (OMG SysML)”, specification, v. 1.3, Object Management Group, Inc., June 2012.
- [6] MagicDraw [computer software], v.18, NoMagic Inc., Allen, TX, 2016.
- [7] Macdonald, M, DocGen (Version X) [command-line documentation tool for software products]; available from <https://github.com/mtmacdonald/docgen>.

BIOGRAPHIES



Oleg Sindiy received a B.S. in Aerospace Engineering from Embry-Riddle Aeronautical University-Prescott in 2004, and M.S. and Ph.D. in Aeronautical and Astronautical Engineering from Purdue University in 2007 and 2010 respectively. Dr. Oleg Sindiy has been a systems architect at JPL since 2011, where he has supported development and operations of variety of space exploration systems such as: CubeSats, ISS instruments, deep space robotic orbiters, landers, and rovers, and human space flight vehicles. He is currently supporting the development of the Information System architecture for the Europa mission’s spacecraft. He was also the initial architect for application of MBSE on the Asteroid Redirect Robotic Mission concept.



Brian Weatherspoon received A.S. degrees in Mathematics, Physics, and Engineering Technology from a combination of Citrus Community College and Pasadena City College in 2016, and has since transferred to California State Polytechnic University, Pomona, where he expects to obtain a B.S. in Aerospace Engineering by 2018. Brian

currently supports the Asteroid Redirect Robotic Mission. He has been a Model-Based Systems Engineer at JPL since January of 2016.



Raffi Tikidjian received his B.S. degree in Computer Science from the University of Cal Poly Pomona in 2003, and went on to receive his M.S. degree in Computer Science specialized in the area of Software Engineering from the University of Southern California (USC) in 2006. He is presently a member of the Mission Control

Systems Engineering & Software Architecture group at JPL. His interests are in the areas of systems health management, simulation and modeling, model based engineering technologies, software methodologies and processes, mobile and web application user interface design. He is currently the lead for implementing Model-based Systems Engineering on the Asteroid Redirect Robotic Mission.



Tanaz Mozafari received a B.S. in Electrical Engineering from Azad Tehran University in 2000, and a M.S. in Engineering Management from California State University, Northridge in 2009. She has been working at JPL since 2010. She is currently the Configuration Management Engineer (CME) lead for the Asteroid Redirect Robotic Mission.