

An Operations Concept for Integrated Model-Centric Engineering at JPL

Todd J. Bayer, Lauren A. Cooney, Christopher L. Delp, Chelsea A. Dutenhoffer, Roli D. Gostelow, Michel D. Ingham, J. Steven Jenkins, Brian S. Smith
NASA Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, California, 91109
Mail Stop 301-230
818-354-5810
Todd.J.Bayer@jpl.nasa.gov

Abstract—As JPL’s missions grow more complex, the need for improved systems engineering processes is becoming clear. Of significant promise in this regard is the move toward a more integrated and model-centric approach to mission conception, design, implementation and operations. The Integrated Model-Centric Engineering (IMCE) Initiative, now underway at JPL, seeks to lay the groundwork for these improvements. This paper will report progress on three fronts: articulating JPL’s need for IMCE; characterizing the enterprise into which IMCE capabilities will be deployed; and constructing an operations concept for a flight project development in an integrated model-centric environment.^{1 2}

TABLE OF CONTENTS

1. INTRODUCTION.....	1
2. ARTICULATING THE NEED.....	2
3. IMCE OVERVIEW	4
4. ARCHITECTING	4
5. IMCE OPERATIONS CONCEPT	7
6. CONCLUSION	14
REFERENCES	14
ACKNOWLEDGEMENTS	14
BIOGRAPHIES	15

1. INTRODUCTION

The mission of the Integrated Model-Centric Engineering (IMCE) Initiative at JPL is to advance our enterprise from the current document-centric engineering practices to one in which structural, behavioral, physics and simulation-based models representing the technical designs are integrated and evolve throughout the life-cycle, supporting trade studies, design verification and system verification and validation. A necessary step toward these ends is to define and agree on the target, i.e., the architecting step. A significant part of architecting is constructing the operations concept: understanding how users would use a model-centric environment to develop future missions.

The IMCE Operations Concept is a work in progress, and this paper reports the status of that work as well as two closely related areas, as summarized below.

First we describe the ways in which the current document-centric processes are failing to keep up with increasing mission complexity. This is the section titled “Articulating the Need”. The goal of this effort is to understand at a sufficiently detailed level where the current flight project development processes are not working well. This has been useful in building understanding of, and advocacy for, the IMCE Initiative. Its main uses however will be for guiding the changes to our present environment and for judging the effectiveness of those changes (the improved, model-centric environment must of course be able to show that these current problems have been fixed).

Next we describe the IMCE initiative and address the notion of architecting and what it means in this context. It is important to realize that IMCE itself is not a system. The system of interest is in fact the JPL Flight Project Development Enterprise. The IMCE Task will deploy capabilities into this enterprise. It is necessary to understand the architecture of this enterprise – both the as-is and the desired, model-centric state.

Finally we describe our approach to, and current status of, the Operations Concept. Infusion of model-centric processes into JPL’s engineering processes will require, among other things, the construction of model-centric analogues corresponding to the current processes, in each technical domain and at each phase of a project’s lifecycle. To accomplish this it is necessary to understand the processes we are attempting to transform. The operations concept will describe key processes which flight projects execute, in a ‘before’ (document-centric) and an ‘after’ (model-centric) view. A key assumption is that JPL’s established engineering lifecycle for flight projects is a mature and successful roadmap. That is, the problems IMCE aims to address are in the ability of flight projects to follow this roadmap, not in the roadmap itself. The approach is to focus on small, almost atomic, pieces of this process, in order to identify the architectural patterns of

¹ 978-1-4244-3888-4/10/\$25.00 ©2010 IEEE

² IEEEAC paper #1120, Version 3, Updated December 11, 2009

interest. Small teams composed of modelers and flight project subject matter experts are developing the use cases. Weekly reviews with the full team as well as two large workshops have provided guidance and feedback.

2. ARTICULATING THE NEED

What problems is IMCE meant to address? At a high level, the core challenge is to better manage growing system complexity. As the character of JPL's missions progresses from fly-by to orbital study to *in situ* exploration, the systems which must accomplish these missions grow in complexity. One of the trends enabling this progression is the ever-increasing processing power of spaceborne computers, which allows more and more of the functionality (i.e., behaviors) of these systems to be implemented in software. Unfortunately, single viewpoint, natural language specifications are inadequate to capture and expose these system level interactions and characteristics (emergent system behaviors). At the same time, the potential range of behaviors has become so large that it is impractical to fully test it: problems can no longer be reliably exposed by testing. **The result of this situation is that inadequately-specified and incompletely-tested system level interaction are a major and growing risk factor for our missions.**

In-flight problems often result from these issues. These problems range in severity from lowered science return all the way to catastrophic mission failures. In the most benign cases, unresolved development issues are left to mission operations to work around, increasing cost and often degrading the science return[1]. In more serious cases significant anomalies occur, some of which are mission-threatening. Escapes in management of complexity have played a clear role in several significant anomalies during the Mars Reconnaissance Orbiter (MRO) mission, as described in [2] and [3].

While real and compelling, these challenges as articulated above are not specific enough to guide the IMCE development. Therefore a study was undertaken to explore in more detail the nature of these challenges, as described below.

Approach to Identifying Current Challenges

The IMCE team looked from several different vantage points at the current way JPL develops flight projects, and identified processes that are broken or functioning sub-optimally. We then looked for patterns and common themes in what we found, and describe them here. This overall approach is guided by a key assumption that JPL's established engineering lifecycle for flight projects, as expressed by the Flight Project Lifecycle, the Gate Product matrix, and the Review Success Criteria, is a mature and successful roadmap. That is, the problems IMCE aims to

address are in the ability of flight projects to follow this roadmap, not in the roadmap itself.

Four themes emerged from this work. We summarize them below, then describe each in more detail with references and examples provided where available (much of the source material is internal, unpublished work). The IMCE Operations Concept will include use cases that explicitly address the main themes here. Note that other themes which are less directly addressed by Model Based Engineering (e.g., training, management) are not addressed here.

Current Challenges: Four Themes

The current challenges in flight project engineering can be grouped into four themes:

- (1) System design emerges from the pieces, rather than from an architected solution, resulting in systems which are brittle, difficult to test, and complex and expensive to operate.
- (2) Knowledge and investment are lost at project lifecycle phase boundaries, resulting in increased development cost and risk of delayed discovery of design problems.
- (3) Technical and programmatic sides of projects are not well-coupled, hampering effective project decision-making and increasing development risk.
- (4) System design re-use is lacking, increasing cost and risk, and damping the potential for true product lines.

Theme 1: System Design Emerges From The Pieces

a. Architectural principles are seldom articulated or used to drive the design. The need for, and the role of, the architect have not yet become a strong part of our engineering processes. Lack of robust architecting allows poorly understood or flawed requirements and assumptions to persist, resulting in weakened designs.

Without strong architectural principles, the quality of the architecture is disproportionately driven by the design process of functional decomposition. But this process alone does not naturally produce a well-architected system, nor does the whole spontaneously re-emerge from the parts. Some of the problem areas arising from weak architecture are: unmanageable interactions, stovepiped analyses, and brittle fault protection, as described below.

Without strong architectural support for system level considerations, the management of ad hoc point-to-point relationships (interactions, interfaces) can become overwhelming.

Simplification for the purpose of analysis does reduce complexity but may overlook interactions. Extensive

decomposing of problems into more “tractable” models can result in conflicting conclusions from sub-models.

Fault protection is largely equivalent to robust design. It needs to be woven into the fabric of a design, where its job is to preserve system functionality over a broad range of conditions -- including faults, but not just faults. Some projects make the mistake of trying to delineate fault protection from nominal functionality to the detriment of both. Such systems tend to be brittle, difficult to operate, and less reliable.

b. Architectural principles, where they exist at all, are sometimes abandoned to solve pressing technical problems.

This is a problem because these “kludges” often make the system more brittle and difficult to operate, increasing costs as well as mission risk. Without stated architectural principles, it is easier to abandon them when the going gets tough. Even when they are stated, as some are in the JPL Design Principles, they may still be abandoned. Only when they are thoroughly and clearly embodied in a design, via the architecture, are they persistent enough to be of value.

c. System designs are spread across many disconnected artifacts (documents, presentations, spreadsheets). It’s not always clear what the approved baseline is. A design change can take months to propagate fully into the baseline design, and the process requires many meetings, word-of-mouth interactions, and emails.

d. Analogously to the scattering of design artifacts, aspects of the design itself are often scattered. Weakly architected systems often result in a situation where many disparate pieces of the design contribute to carrying out a given function with little high-level coordination. The control parameters for these functions are similarly scattered throughout the system, and are therefore more difficult to understand, let alone manage. This problem is exacerbated by the sheer numbers of control parameters (e.g., more than 20,000 on MRO).

e. Physics-based models in the various engineering specialties are not connected to each other or to a system model. Trade studies and system analyses require manual integration of results among all these domains. The inefficiencies make trades and other analyses take longer than necessary, and make examination of more than a few point designs impractical. This ‘stovepiping’ of analyses can also hide significant system level interactions which may only be ‘discovered’ during system test or, worse, after launch.

f. Insufficient consideration of verification and validation (V&V) during requirements development can render aspects of the design untestable, increasing mission risk.

g. Science requirements are not well coupled to flight and ground system requirements, causing the actual ‘goodness’

of a given engineering solution, in terms of its science return, to be unknown until it’s too late to change the design. The inadequate quantitative understanding of how changes in engineering parameters affect the science objectives (the so-called science merit function) can result in lost opportunities to make risk/cost/schedule saving trades, and can even result in a system which does not satisfy the science objectives.

h. Desired system behaviors are poorly articulated by the current textual representation. Using these textual representations, system engineers are unable to communicate meaningfully with software engineers about the desired system behaviors. This results in systems whose behavior must be ‘discovered’ – often in flight.

Theme 2: Knowledge & Investment are Lost Across Phases

a. Models used during the formulation phase of a project are abandoned and new ones created when implementation phase begins. Many examples exist: trade models, cost models, power, telecom, data transport models.

b. Configuration Management (CM) of existing models is lacking, impeding continued use in the next phase (or re-use on the next project).

c. Essential attributes of design are not captured consistently in a readily accessible manner. Architectural principles, Trade study assumptions and rationale, and system design, rationale, and narrative are only partially captured at best and are seldom readily accessible.

d. Training of engineers joining a project takes longer than necessary. Moving from pre-phase A to Phase A, and then to Phases B, C and D, significant team expansion and turnover occur. Training people who were not around for previous phases is currently heavily dependent on getting key documents from, and having lengthy conversations with, key people. Because the system design is poorly captured, this essentially ensures that new team members will be discovering attributes of the design for years.

Theme 3: Technical & Programmatic are Poorly Coupled

a. The cost, schedule, scope, investment and risk implications of a given set of requirements, science objectives, components and functions is very difficult to determine. Very little coupling exists between the performance side (science, requirements, components, and functions) and the implementation side (implementation, investments, and risks). Schedule and cost models rarely include mitigations/backup options for technical implementation risks. It is difficult to transfer information between tools and different discipline types.

b. Trade studies seldom fully incorporate programmatic considerations. There is an unfilled need to be sure that systems engineers are knowledgeable about the

programmatic realities of a project and the impact of engineering decisions on programmatics. The existing tools do not support such an integrated view.

Theme 4: System Design Re-Use is Lacking

a. Because system architectures and designs are not well-captured, re-using them on subsequent projects is difficult and seldom happens (except where the project team itself is ‘inherited’ by the next project). There is currently no formal way to document and integrate the broad experience and knowledge of engineers across a project, to train new systems engineers who will need to absorb this broad knowledge quickly and deeply, and to make this available as a legacy to future projects in a manner that is easily dissected and reapplied to new circumstances. The current institutional guidance (e.g., JPL Design Principles), while providing important and useful heuristics and lessons learned, is not sufficient to enable architecture re-use.

3. IMCE OVERVIEW

The objective of JPL’s Integrated Model-Centric Engineering Initiative is to advance engineering practice to a state in which descriptive and analytical models representing technical designs and relating them to stakeholder concerns are developed and integrated throughout the mission life cycle, from early concept through operations. The IMCE team encourages and facilitates this transition by providing training, user community support, tool/model integration, modeling standards, and a reusable models repository, in partnership with engineers, flight projects, and institutional management.

The IMCE Initiative is beginning its second year. The emphasis in this early phase is development of initial capability (personnel and process in addition to technology) and demonstrating value in order to encourage uptake and adoption of model-based engineering practices.

In recognition of the growing importance of the Systems Modeling Language (SysML) [4] as an industry-wide standard, the IMCE Initiative sponsors semiannual training in SysML and the use of SysML modeling tools. In addition to formal training, there is an active and growing Modeling Early Adopters group that meets informally to exchange problems and ideas.

The bulk of the IMCE workforce at this stage of the initiative is devoted to developing the modeling environment, which consists of tools, repositories, standards, conventions, and guidance, in addition to consulting services. Particular focus is given to practices that support and encourage building interrelated models that cross system, organization, and life-cycle boundaries. For example, subsystem modelers should be able to relate their design elements to system design elements authorized by a

different project work element, created by a different organization, in an earlier life-cycle phase. This interrelatedness is a key feature of the *integrated* nature of IMCE. Achieving it requires, among other things, standards for naming and classification of model elements and properties (ontologies) and the expression of those standards in SysML-specific terms (profiles). It also requires established conventions for model organization and configuration management (including access controls).

Another important aspect of the IMCE Initiative is explicit integration with and support for JPL’s established engineering life cycle. For example, our life cycle defines a set of milestone reviews, with defined scope, deliverables, and success criteria for each. The IMCE team is analyzing these definitions in order to identify model-based deliverables (or deliverable elements) and artifacts that may be applicable evidence for assessing success. Any such standard deliverables and artifacts will then become candidates for development of generator applications built upon a common model access, analysis, and product generation infrastructure. Through these developments the IMCE team intends to systematize and automate key elements of the engineering life cycle at JPL, with the benefits of improved information richness and consistency at lower cost.

The IMCE Initiative has supported several modeling pilot projects, the purpose of which has been to gain experience in the use of SysML and SysML tools, to gather requirements for institutionally-supported modeling infrastructure, and to validate IMCE solutions to enterprise modeling challenges (e.g., configuration management). The pilots have focused on existing JPL projects, namely Gravity Recovery and Interior Laboratory (GRAIL), Soil Moisture Active/Passive (SMAP), and the proposed Jupiter Europa Orbiter mission.

4. ARCHITECTING

The IMCE Initiative is currently in the architecting stage, but what exactly is it architecting? It is important to realize that IMCE itself is not a system. The system of interest is actually the JPL flight project development enterprise. Our objective is to improve the effectiveness of this enterprise through the infusion of model-centric capabilities. In order to accomplish this objective, it is necessary to address the current and future architectures of this enterprise.

The current JPL flight project development enterprise has an architecture—every system does³. This architecture,

³ We adhere to the definition of architecture from the IEEE standard 1471–2000, “Recommended Practice for Architectural Description of Software-intensive Systems”, which is now also ISO/IEC 42010:2007 [5]: the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution.

etc. The envisioned architecting process is illustrated in Figure 2 and described below:

- Produce a specification of the current architecture, consisting of a set of general engineering application patterns, captured in a set of useful views. The process of constructing these models will reveal “pain points”, risk areas, and inefficiencies in our current architecture and processes, representing improvement target. These patterns will be produced by examining and abstracting views from a representative set of engineering applications (see Figure 2). Examples of representative engineering applications are discussed in Section 5 of this paper.
- Once the current architecture has been adequately specified, the evolution of this architecture from its current state to the desired future state needs to be specified. This evolution will necessarily consist of a sequence of incremental steps, in which manageable changes are documented through similar views and then implemented.

This architecting process is currently underway at JPL for our flight project development enterprise. Key questions that need to be answered include:

What “views” do we need?

As mentioned above, answering this question involves identifying the key stakeholders and their concerns/needs,

and deciding what viewpoints (and specific views) would be of most utility in addressing those concerns. In addition to the above-mentioned operational viewpoint (including views captured in use cases and activity diagrams/descriptions), we anticipate needing viewpoints that capture information and data flow through our systems; the set of engineering tools used and the relationships between them; workflow for intra- and inter-organizational collaboration; inheritance and reuse of models and software; and others.

What methodology(ies) do we follow to produce these required views?

At this point, we have only a partial answer to this question. We will undoubtedly use some combination of our current engineering practice (domain-specific thinking, via engagement of the appropriate domain experts at JPL), and aspects of model-based engineering methodologies that resonate with our architecting and engineering teams (e.g., our in-house developed State Analysis methodology [6], the Object-Oriented Systems Engineering Methodology [7], the Rational Unified Process [8], etc). The question of methodology is a critical one, and work is underway to answer this question.

How do we train our people?

In this case “training” involves both the people who need to produce these views (the architecture team), as well as the people who will need to do their work within the context of the (re-architected) enterprise (the JPL engineering community). JPL’s IMCE initiative is beginning to address

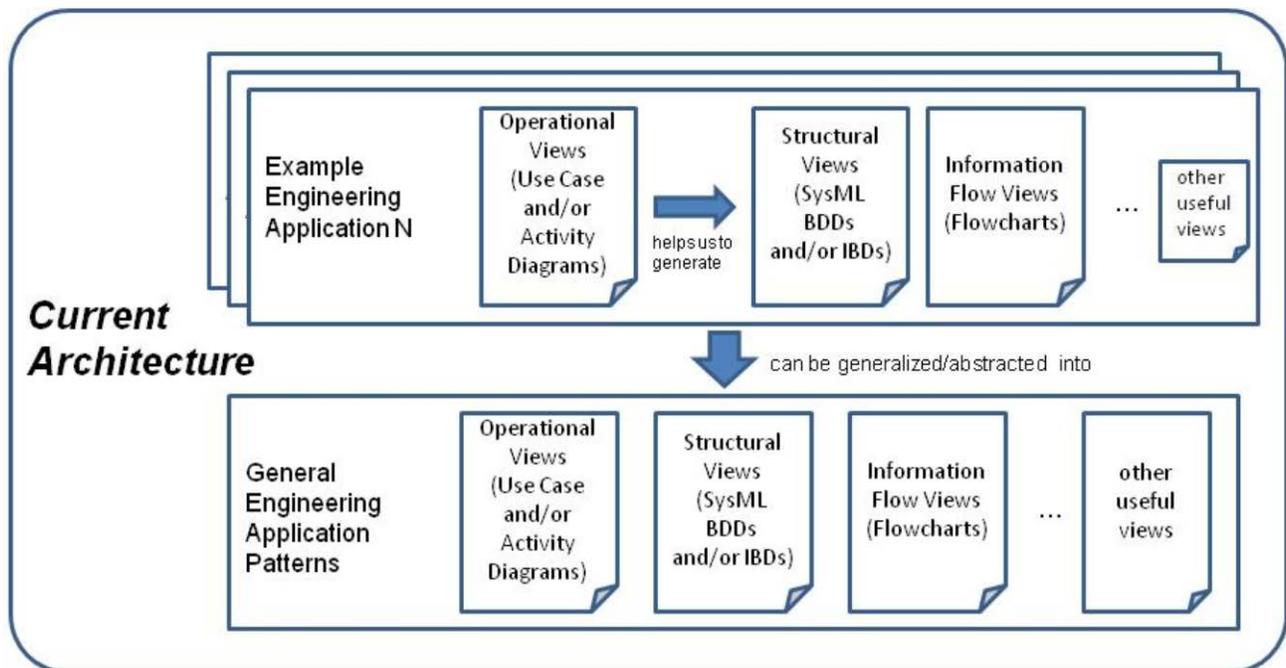


Figure 2 Documenting the current structure of our Flight Project Development Enterprise.

this question in two ways: (1) by sponsoring training classes in the areas of system and software architecture and model-based engineering (for both systems and software) methodologies and representations, and (2) by providing leadership, support and resources to the growing community of practice in model-based engineering at JPL.

5. IMCE OPERATIONS CONCEPT

Development Approach

The approach begins with capturing user scenarios in support of use-case development (Figure 1). The team selected several Subject Matter Experts (SME) in Flight Systems development to review trade studies and requirements development processes. Capturing as-is models of workflows would help identify specific problems that an integrated model-centric environment could solve.

We resolved to “eat our own dog food” and capture these workflows and architectural views in a SysML model: to be successful with furthering the adoption of modeling we need to adopt modeling ourselves as a demonstration of the effectiveness and utility we are advocating. In other words – if modeling can’t be used to solve the problem of designing IMCE then how could it solve anything else?

The approach of low-level workflow models has become an important aspect of our approach. Based on the work so far, the root causes of many of the problems identified in section 2 are thought to be deeply and pervasively embedded in low-level interactions among people and tools. So, to discover and mitigate these root causes, we must look at these low level interactions. This approach is consistent with the assumption previously stated that the high level process (i.e., the overall engineering lifecycle) is sound.

SME insights into these workflows are key to a complete understanding of the as-is architecture. It was assumed that past lessons learned have already been incorporated into the current practices. It is critical not to lose this experience when developing a model-based approach to systems engineering. By conducting these studies with a few different engineers who were experienced in each area, the modeling teams assumed a fair sampling of existing knowledge. The as-is models were reviewed with the system engineers whose experience had been used to build them to ensure accuracy. It is important to note that this is supplementary to the eventual incorporation of institutional best practices, rules, processes etc – much work is still required.

To ensure consensus on the work flow feed back from the SMEs, model reviews were also held with members of the model-based community to review and help interpret and guide the modelers’ interactions with the SME’s.. These experienced modelers also had a variety of different systems

engineering experience, and helped mainly by bringing to bear a depth of knowledge and experience from past modeling endeavors. The models were also presented to larger audiences at two IMCE workshops.

Basic Process for Information Gathering

The first step in the process was to select a set of activities on which to focus. Ideally, this selection would largely cover the spectrum of activities performed within the organization. However, time constraints, as well as an awareness of the diminishing returns on repeating the process, limited the number of activities explored so far. The reason that diminishing returns exist is that there are fundamental issues and needs that will be uncovered multiple times, while the number of new issues and needs left to find decreases with each new study performed. Therefore, two activities were selected: Trade Study Performance and Requirements Development in Phase A. Phase A is a significant point to start these activities as it is the hinge-point between formulation and implementation – the most problematic point of disconnect in the current enterprise. It also positions us for application of this work to the proposed Jupiter Europa Orbiter mission. A team of two people was assigned to each activity. The teams followed approximately the same steps to achieve the goals outlined in the Approach section above.

The basic steps performed by each team were:

- (1) Identify and interview stakeholders and subject matter experts
- (2) Develop a high-level workflow and dataflow model of the current process by which the activity is performed
- (3) Identify an individual sub-activity within the process to expand in more detail
- (4) Develop a detailed workflow and dataflow model of the sub-activity as it is currently performed
- (5) Develop a detailed workflow and dataflow model of the sub-activity as it might be performed in a model-centric engineering environment
- (6) Analyze models developed in steps 4 and 5 to discover needs

At each step, a panel of experienced systems engineers reviewed the results for correctness, completeness, feasibility, and accessibility to the intended audiences. Therefore, the completion of each step involved several iterations.

Detailed Process Description

Step 1 – Identify and interview stakeholders and subject matter experts

In order to model the workflow and dataflow of performing a trade study, the team needed to understand the process. The team gained this understanding through interviews with four subject matter experts and stakeholders in the area of Phase A Trade Studies. The dialogue that took place in these interviews was translated into SysML workflow and dataflow diagrams. The first draft of this was reviewed by two of the subject matter experts and revised based on their comments.

Step 2 – Develop a high-level workflow and dataflow model of the current process by which the activity is performed

This step was initially intended to be a main product of the study. However, as time progressed, it became clear that its function was rather to provide context for a deeper analysis. In either case, the correctness of this model was of importance. In order to try to ensure correctness, the modeling teams held interviews with several experts on the study topics of differing backgrounds and focus. The modeling teams used these discussions to generate models to describe the process flow and the dataflow of the entire activity. These subject matter experts reviewed these diagrams and made suggestions and revisions, which the modeling teams used to develop a second version of the diagrams. The panel then reviewed these over the course of several meetings in order to arrive at the final models.

Step 3 – Identify an individual sub-activity within the process to expand in more detail

As the review of these products continued, it became clear that they were not expressed at a low enough level of detail to enable the main goal of helping define and elucidate the Concept of Operations. This is due to the fact that there are no major issues with the high-level process: it has evolved over time to meet the needs of the project at each phase and cannot be altered without failing to address the needs it was developed to meet. However, what areas do present problems to be solved are the means by which each step within the whole process is completed. Therefore, it was decided that each team would select a sub-activity within the process to expand in more detail in order to identify those problems and posit solutions to them. Again, this process would ideally be completed for each sub-activity, but due to various constraints, each modeling team analyzed only one sub-activity so far. The sub-activities chosen were selected because they are critical steps in the high-level process.

Step 4 – Develop a detailed workflow and dataflow model of the sub-activity as it is currently performed (As-Is)

Once each team chose the sub-activity of focus, they constructed models that describe what currently happens (as-is) inside the sub-activity. Modeling at this level was productive because it was at a sufficiently detailed level to enable problems to become apparent, but not so low as to be cluttered with detail at the level of mouse-clicks or

individual conversations. Like the previous models, these describe both the actions performed, and the dataflow associated with each activity. The models were reviewed by the panel and refined several times.

There was a difference between how this step was completed by the Trade Studies Performance modeling team and the Requirements Development modeling team. The Trade Studies modeling team found it useful to also create a logical model at the low-level that is “method-independent”. They then described the steps of the current process (as-is) as occurring within these logical steps. This made it easier to compare the current process with a model-centric engineering process by introducing conceptual divisions into the process model. This approach worked well for Trade Studies modeling team because the logical steps and basic ordering for performing a trade study is clear. The Requirements Development modeling team found it more difficult, because the highly interdependent nature of the steps in that process makes it less productive to create conceptual divisions between steps.

Step 5 – Develop a detailed workflow and dataflow model of the sub-activity as it might be performed in a model-centric engineering environment

Using the same sub-activities as were used in Step 4, the teams created models that describe how that sub-activity might be performed in a model-centric way. Thus, this step involved both modeling the process and also brainstorming to come up with the potential processes. Any problems that were apparent in the as-is models were taken into consideration in defining the model-centric processes. The models were reviewed by the panel and refined several times. The Trade Studies team again used the logical grouping to help organize the process. Like the models produced in Step 4, these descriptions focused on the level of detail just above minute-by-minute behaviors, and also contain dataflow information.

Step 6 – Analyze models developed in Steps 4 and 5 to discover needs

The modeling teams used the models created in Steps 3 and 4 to identify problems in the existing processes (both in terms of workflow and dataflow). They could then verify that they were mitigated or obviated in the model-centric processes. They also used the models to identify any capabilities added in the model-centric process that are not present in the current process. These gave confidence in the usefulness of the model-centric processes. The completion of Step 5 also explicitly drew out a set of capabilities that IMCE should provide and some architectural features and infrastructure that could enable those capabilities. These will be used to help understand the needs of stakeholders and develop the requirements for IMCE, as well as to define and describe the system in the Concept of Operations.

Discovery:

Proto-Principles and Guidelines

Principles are general rules and guidelines, intended to be enduring and seldom amended, which inform and support the way in which an organization sets about fulfilling its mission. In order to provide guidance to development of the Concept of Operations, we have begun to identify architectural principles and properties of concerns of stakeholders.

The guiding principles identified in this work largely revolve around enabling more sophisticated modeling and data access while minimizing the invasiveness into the user space. Although the team has not formally adopted such principles yet, some initial development principles have been captured and provisionally used:

- IMCE should add value to current investments in modeling
- IMCE should not get between the users and their tools
- IMCE should not require overnight change
- IMCE should make it easy to do the right things, and difficult to do the wrong things
- Models should be re-usable and evolvable
- Information may exist in multiple locations, but all information should have one locatable 'truth' source
- Artifacts should be machine-readable
- There should be no us (IMCE developers) and them (IMCE users)
- Engineers must be able to trust modeling and development tools.

Workflow Analysis Discoveries

In our study we found that, when a single user is responsible for maintaining their complete view of the system design, information and accuracy are lost both in data transfer and due to the isolated nature of the analysis. We investigated how a user might perform trade studies and develop requirements in a model-centric environment, focusing on actions performed by the SME and how the model would fit in to the framework.

Guided by our emerging development principles we outlined some workflow views of how modeling and simulation could take place in a more integrated environment. These concepts provide the seeds for the conceptual architecture from an operational viewpoint. Generally, we discovered that engineering and design processes have a lot of creative and ad-hoc activities. These activities no doubt promote

good design, however they rely on things like email and office productivity tools to express complex and sophisticated design concepts and decisions. These tools lack the capability to capture detail and expressive semantics in a scalable way. Interconnection between sets of information is also a highly manual process—there is nothing which checks if the value of a given property is the same in two different documents.

There needs to be a single authoritative source for each piece of data used in the model. As envisioned, there should be a manual mapping of model parameters to data source parameters. Once the engineer has performed this mapping, the computer maintains this relationship and tracks metadata about configurations. Users waste a lot of time collecting the data, and in fact that data may be out-of-date, out-of-context, or incorrect due to clerical errors. The model provides a map for where to find the authoritative source for that data. Figure 3 shows the high-level view of a trade study process in a model centric environment.

Throughout the project lifecycle, there is substantial knowledge loss due to information spread between a wide range of documentation and non-computer readable formats, such diagrams in presentation slides. The fact that the model itself contains metadata about the design provides comprehensive capture about methods, initial conditions and results. This allows for complete documentation and information capture/querying capabilities.

For example, in the model-centric trade study information and meta-information about the trade study would be available through a registry. Such a capability would allow for the capture of what information the trade used, rationale and outcomes in a single source of truth.

Often information (documents, requirements etc.) and simulations are reused from previous projects. This practice is highly desirable but it can be difficult to gauge what is appropriate to reuse. Modeling environments can facilitate this to a much better extent. Investments in libraries, templates, profiles, ontology and modular simulations will foster reusability.

For example, talking to some of the SMEs, it was discovered that in the current process a lot of analytical models are created for a specific purpose, used for that purpose and then no longer maintained. Figure 4 shows this process where data from one model is hard-coded into another model designed for a specific use. This is a problem, because if something changes and the analysis needs to be redone, the model is often so out-of-date that it's easier to make a whole new model than to reuse the existing one. In the model-based paradigm (Figure 5), the model is maintained throughout the entire lifecycle. Instead of abandoning preliminary models used in early development once the requirements are established, the preliminary system model can be refined and specialized as the design

matures. In contrast to many existing ad hoc methods, the inherent scalability of a SysML description allows the model to grow and evolve as detailed design information is added. Also, existing simulation tools can be tied in with the system models to get a richer and more complete view of the system than the current methods allow.

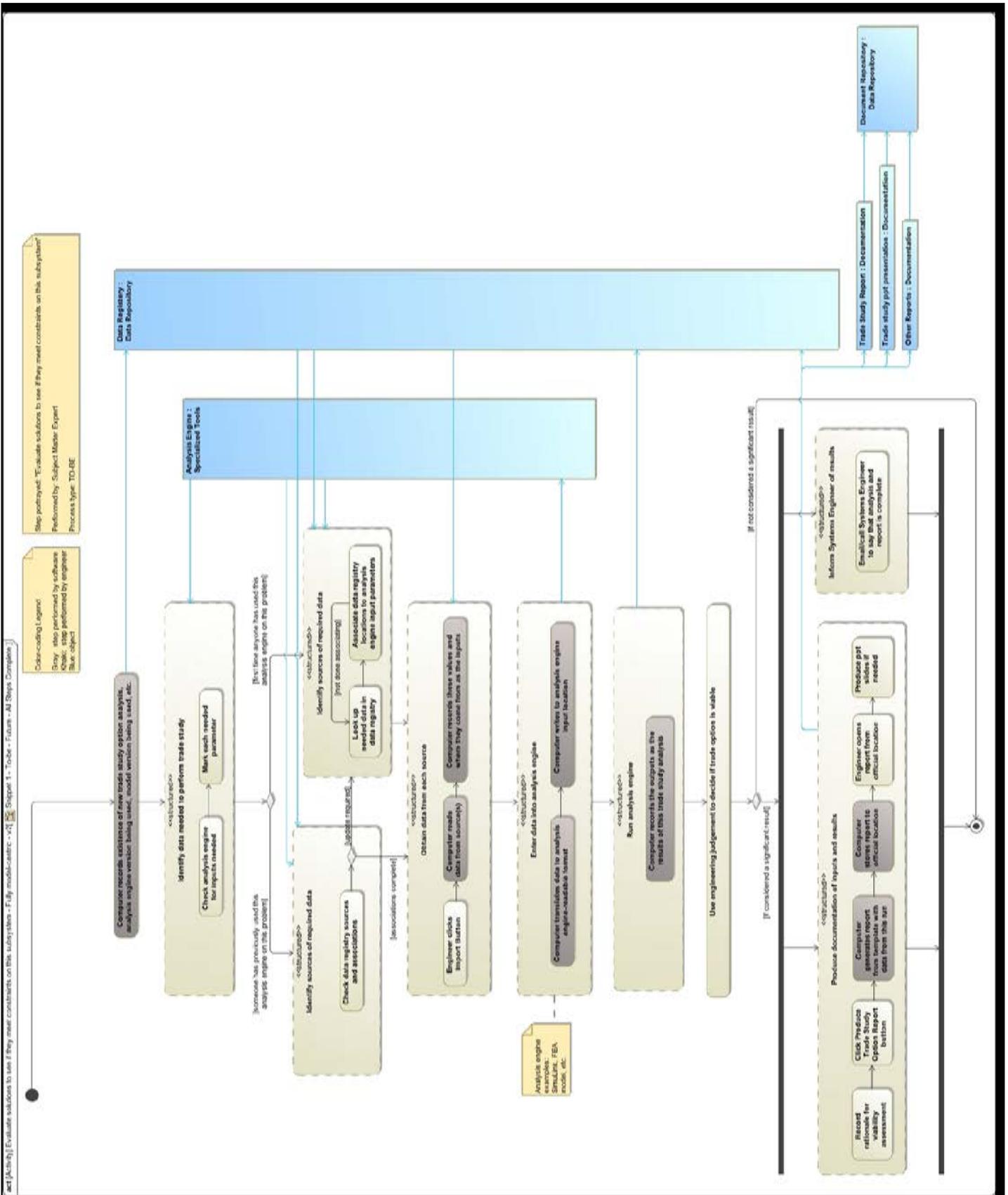


Figure 3 High Level View of Trade Study Process in a Model-Centric Environment

would automate this process, as well as capture these transactions and verify their correctness, completeness, etc.

Information that is passed between different engineering roles is informally tracked and could be captured in a way that is more transparent to the user. This avoids the use of emails as a means of cataloging design information, a task for which email is poorly suited. Similarly, when the requested information is received, which could take some time depending on how significant a request it is, it can be checked against the request and additionally verified for other forms of completeness like units, etc.

For example, Figure 6 shows manual model conversion performed by multi-body dynamics engineers and finite element analysis engineers. These engineers convert a design model generated from a specific tool (“NX”) to two different formats, each to be used by two separate tools (“ADAMS” and “Femap”). Such model transformations are

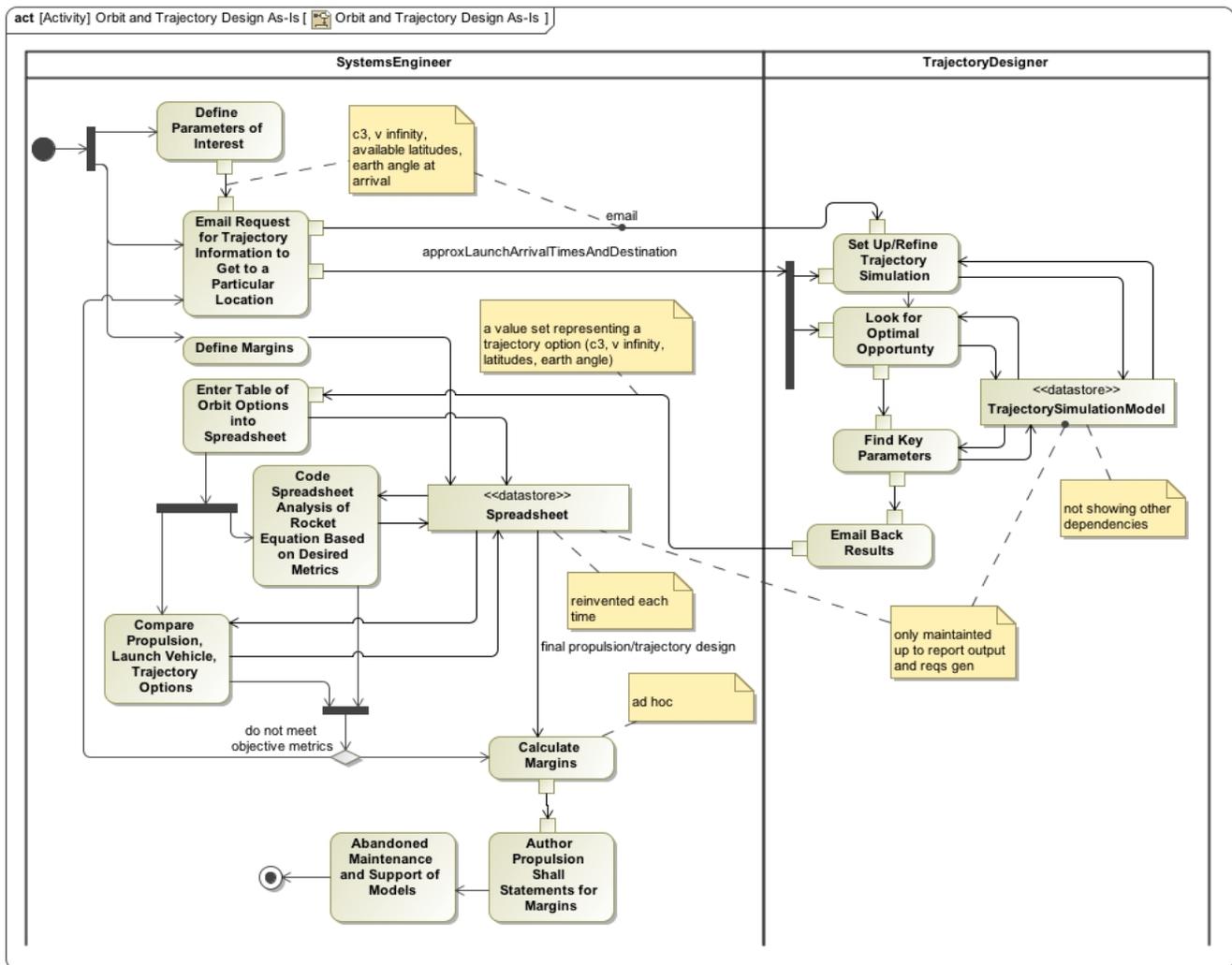


Figure 4 Ad Hoc Models During Requirements Development

time consuming and tedious. A model centric environment

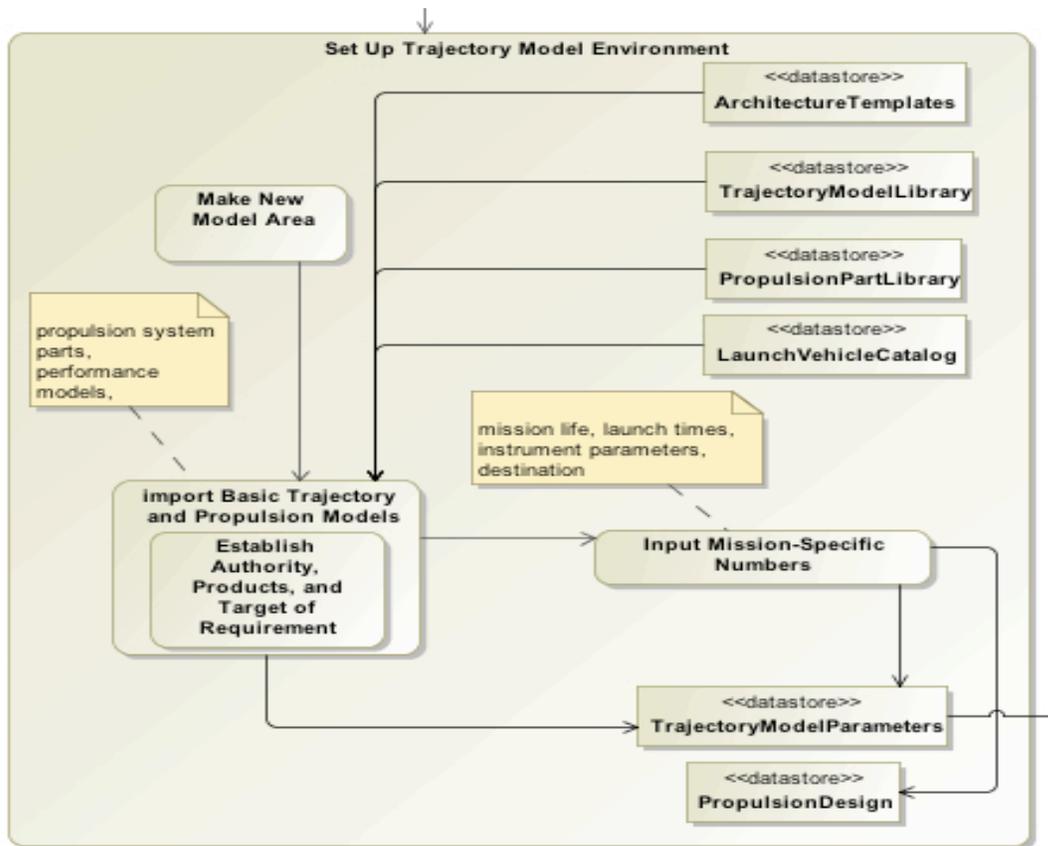


Figure 5 Model Re-use Concepts to Support Requirements Development

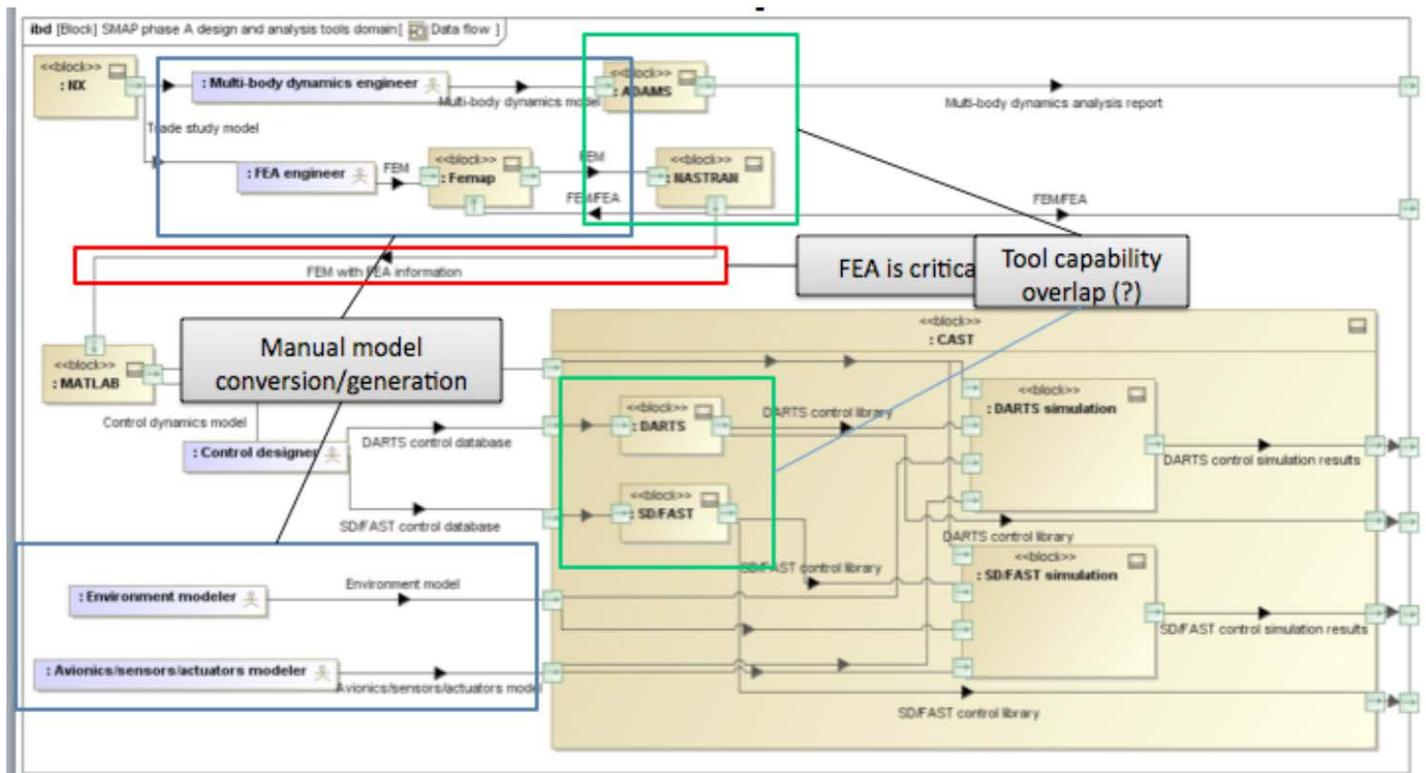


Figure 6 Patterns of Information Flow in Current Architecture

6. CONCLUSION

JPL's current engineering processes suffer from discontinuities at major lifecycle phase boundaries, between systems and subsystems teams, between technical and management domains, and from one project to the next. These discontinuities increase cost, stretch schedules, and increase both development and mission risk. The goal of the IMCE Initiative is to facilitate an evolution to a more model-centric enterprise which should contribute to mending these gaps.

We have begun the work to understand the scope of such a transition. In the context of system architecting, this will involve delivering targeted, enabling capabilities into the JPL flight project development enterprise. We are working to describe the current architecture of this enterprise, and how it should be evolved. The IMCE Operations Concept is a key piece of this work, aimed at elucidating the concerns of the users of these capabilities. Work on the Operations Concept is well underway and is already yielding valuable insights.

During the coming year we plan to complete the operations concept work and provide the results in a form useful for architecting the needed changes to the flight project development enterprise. Major elements of the work to go include: modeling additional relevant scenarios in other life cycle phases and to cover other areas of concern such as model building/reuse; studying our results to discover the essential patterns of the current architecture, how they should change in a model-centric environment, and what high level requirements are implied; and deliver all of this to the IMCE team as a fully-documented SysML model.

It's been said that nothing succeeds like success. A corollary might be that nothing raises expectations for future success like past success. Each successful mission of discovery answers important scientific questions. It also creates many new questions, which are often more difficult to answer, requiring more capable observing platforms and a closer vantage point. We must equip ourselves with more powerful tools and methods if JPL and its partners are to continue rising to these challenges.

REFERENCES

- [1] "Managing the On-Board Data Storage, Acknowledgement and Retransmission System for Spitzer", Marc Sarrel, Carlos Carrion, Joseph Hunt, Jr., 2006, AIAA Paper 2006-5564
- [2] "In-Flight Anomalies and Lessons Learned from the Mars Reconnaissance Orbiter Mission", Todd Bayer, 2007, IEEEAC Paper #1451
- [3] "In-Flight Anomalies and Lessons Learned from the Mars Reconnaissance Orbiter Mission – an Update", Todd Bayer, 2008, IEEEAC paper #1086
- [4] Object Management Group. OMG Systems Modeling Language (OMG SysMLTM), Version 1.1. November 2008.
- [5] IEEE Std 1471-2000. IEEE Recommended Practice for Architectural Description of Software-Intensive Systems.2004.
- [6] Ingham, M., Rasmussen, R., Bennett, M., and Moncada, A., "Engineering Complex Embedded Systems with State Analysis and the Mission Data System", AIAA Journal of Aerospace Computing, Information and Communication, Vol. 2, No. 12, Dec. 2005, pp. 507-536.
- [7] Sanford Friedenthal, Alan Moore, Rick Steiner, "A Practical Guide to SysML, The Systems Modeling Language" (The MK/OMG Press). August 2008.
- [8] The Rational Unified Process: An Introduction, 2nd ed., by Philippe Kruchten (Addison Wesley Longman, 2000)

ACKNOWLEDGEMENTS

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

The information in the section on Articulating the Need was drawn from many internal unpublished sources, including works by Mark Brown, John Casani, Steve Cornford, Frank Deakens, Jennifer Mindock, Brian Muirhead, Robert Rasmussen, and Marc Rayman. The authors wish to acknowledge their significant contributions to the IMCE effort.

BIOGRAPHIES



Todd J. Bayer is the Assistant Manager for Flight Projects of JPL's Systems and Software Division where, among other pursuits, he is working to make integrated model centric engineering a reality at JPL. He received his B.S. in Physics in 1984

from the Massachusetts Institute of Technology. He started his career as a project officer in the US Air Force at Space Division in El Segundo, California. Following his military service, he joined the staff of JPL in 1989. He has participated in the development and operations of several missions, including Mars Observer, Cassini, Deep Space 1, and Mars Reconnaissance Orbiter. During a leave of absence from JPL, he worked as a systems engineer on the European next generation weather satellite at EUMETSAT in Darmstadt, Germany. He was the Lead Flight System Engineer for MRO development, and Chief Engineer for MRO flight operations.



Christopher Delp is Architectural Engineer for Multi-Mission Operations and a member of of the Flight Software Systems Engineering and Architectures Group at the Jet Propulsion Laboratory. His research interests include software and systems architecture, model-based systems engineering applications, and

real-time embedded software engineering. He earned his M.S. and B.S. degrees from the U of A in Systems Engineering.



Lauren Cooney is a Systems Engineer in the Guidance, Navigation and Control Section at JPL. Her interests include control, autonomy and intelligence for space and underwater systems. She received her SM in Ocean Engineering and SB in Mechanical Engineering from MIT.



Chelsea Dutenhoffer is an engineer in the Systems and Software Division at JPL. She is currently the Ground Data System Engineer for the Dawn project and also performs testing on multi-mission ground software used at JPL. Her primary interest is how model-based systems engineering practices can make complicated

space systems easier to understand and manage, especially ground data systems. She holds a B.S. in Aerospace Engineering from Embry-Riddle Aeronautical University.



Roli Gostelow is a Software Systems Engineer in the Integrated Ground Data section at JPL. She is currently involved in software quality improvement, web design, and model-based systems engineering work. She holds a B.S. in Computer Science and in Mathematics from the University of Chicago.



Michel Ingham is the technical group supervisor of the Flight Software Systems Engineering and Architectures Group at the Jet Propulsion Laboratory. His research interests include model-based methods for systems and software engineering, software architectures, and spacecraft autonomy. He earned his Sc.D. and S.M. degrees from MIT

in Aeronautics and Astronautics, and a B.Eng. in Honours Mechanical Engineering from McGill University in Montreal, Canada.



Steven Jenkins is a Principal Engineer in the Systems and Science Division at the Jet Propulsion Laboratory, currently supporting JPL's Integrated Model-Centric Engineering Initiative. His interests include application of semantic and modeling technologies to systems engineering. Dr. Jenkins holds a B.S. in Mathematics from Millsaps

College, an M.S. In Applied Mathematics from Southern Methodist University, and a Ph.D. In Electrical Engineering (Control Systems) from UCLA.



Brian Smith is a Software Engineer in the Flight Software Systems Engineering and Architectures Group at the Jet Propulsion Laboratory (JPL). His research interests include autonomous systems, robotic systems, and the verification and validation of flight software systems. He earned his Ph.D. and M.S. degrees in Electrical

and Computer Engineering from the Georgia Institute of Technology, as well as his B.S. degree in Computer Engineering. Dr. Smith is also currently developing and testing software for the Mars Science Laboratory (MSL). While at Georgia Tech, Dr. Smith designed and implemented the user interface and control software for a multi-robot network for Antarctic climate research, as well as implementing path planning software for unmanned helicopters.

