Model-Based Engineering Design Pilots at JPL

Mark Kordon, Steve Wall, Henry Stone, William Blume, Joseph Skipper, Mitch Ingham,
Joe Neelon, James Chase, Ron Baalke, David Hanks, Jose Salcedo, Benjamin Solish, Mona Postma, Richard Machuzak
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
818-393-0476
Stephen.D.Wall@jpl.nasa.gov, Mark.A.Kordon@jpl.nasa.gov

Abstract—This paper discusses two recent formulation phase Model-Based Engineering Design pilot projects at the Jet Propulsion Laboratory. It describes how model-based functional and state analyses were synthesized and integrated with system performance simulation and mission planning then piloted in the formulation phase of two deep space missions.

TABLE OF CONTENTS

1.0 INTRODUCTION

At the Jet Propulsion Laboratory (JPL) the life-cycle of a deep space mission normally goes through six phases, each culminating with a review by project management and its funding agencies [1]:

- **Pre-Phase A:** Advanced Studies
- **Phase A:** Mission & System Definition
- **Phase B:** Preliminary Design
- **Phase C:** Design & Build
- **Phase D:** Assembly Test & Launch Ops
- **Phase E:** Operations

Model-based engineering design (MBED) enhances the description of a system beyond traditional language, utilizing both descriptive and simulation models to improve the understanding and verification of the system.

In fiscal year 2004 (FY04), JPL's Research and Technology Development (R&TD) program undertook a multi-year initiative to incrementally build an end-to-end, concept to operations, model-based approach for JPL space mission system design. In FY06, the initiative focused on the spacecraft (S/C) formulation in Phase A. Since many of the activities in this phase center on system engineering, the initiative focused on developing its model-based system engineering (MBSE) capabilities.

Systems engineering has been described as an interdisciplinary engineering management process that evolves and verifies an integrated, life-cycle balanced set of solutions that satisfies customer needs [2]. Rather than focus on one particular aspect of the system, system engineers address broad system-wide issues such as cost, schedule, risk, performance, training, support, test, manufacturing and operations throughout a project's lifecycle.

Early in a project, system engineering tasks focus on defining and documenting customer needs, desired functionality, and requirements. Later as the system is being developed the emphasis shifts to design synthesis and system validation. At JPL these activities span many domains including project system engineering, flight system engineering, payload system engineering, software system engineering, and mission operations system engineering, in an attempt to meld each technical discipline into a unified whole.

The lab's current system engineering methodology was adapted from the document-based paradigm established in the 1940's and 50's by the Manhattan project and early

[1] 1-4244-0525-4/07/$20.00 ©2007 IEEE.
[2] IEEEAC paper #1678, Version 11, Updated December 11, 2006

space program engineers. While this approach has been responsible for many successes, early space projects were relatively simple when compared to recent missions. It is now widely recognized that the document-based approach does not scale well. There are a number of well-documented deficiencies when applying these techniques to larger, more complex systems, most notably in the areas of requirements traceability and verification and validation (V&V) activities.

Recent missions have required extremely precise measurements from highly sensitive instruments in increasingly remote and unfamiliar environments. This impacts the entire system, requiring complicated and/or physically large mechanisms with many interconnected and compounded processes. With many more interdependencies, the chances of a requirement or design change having an unintended and possibly catastrophic effect on the project increases. Being able to quickly recognize and react to unanticipated effects and/or budget issues is critical in JPL's schedule and cost constrained environment.

MBSE is an alternative to the current document- based approach. MBSE analysis and design methods utilize structured, graphical, implementation-independent notational systems for producing unambiguous documentation of system requirements and design. As with other notational systems these model-based languages have syntax (structure) and semantics (meaning). What makes them unique is that they can produce artifacts that are both machine-readable and human-readable. This means that consistency, completeness and integrity of requirements and design can be checked with automated system audits. It also means that one can select and quantitatively transform system engineering information, enabling other kinds of analyses such as simulation-based risk and performance assessments.

By adopting a model-based systems engineering approach, models become the project knowledge, capturing all rationale and decisions. Requirements and designs are stored and exchanged using a standardized information model. The expressive power of models and the rigor of a model-capture and analysis process offer great benefits. Much like the advance that algebraic notations offered physics; the model-based approach provides clear definitions of behavior, capability, and design for system engineering activities.

Computer-based simulation models, on the other hand, are commonplace in academia and industry. Supported by increasingly powerful commodity computer hardware and modern computational methods, simulations are able to provide better forecasts faster than ever before. For this reason they are now indispensable tools in many business, engineering and science disciplines. Aerospace companies, for example, use simulations to analyze and tune vehicle performance, while automobile manufacturers use them as a means to upgrade manufacturing processes and simulate crash tests to improve vehicle safety.

Simulations are widely accepted and used in investment analysis, decision (risk) analysis, systems analysis, product design, and training. In engineering applications, simulations enable early detection of subtle design errors allowing engineers to reliably create complex systems that are well beyond the capability of traditional approaches. At JPL they are often important tools in minimizing risk and maximizing flight readiness.

Integrating the model-based systems engineering process with the simulation and mission planning processes is an important aspect of model-based engineering design. This type of integration is different than the time-based integration needed in simulation models. In process integration, processes (or applications) are run to completion and their data is passed to other processes. Thus, the focus of process integration is inter-application communication and execution rather than event handling.

The paper begins with a brief overview of the traditional functional analysis approach and a newer state analysis approach to systems engineering. It then goes on to discuss the synthesis of these two approaches, and their integration with performance simulations and mission planning. The paper concludes by presenting the results of two JPL formulation phase pilot efforts.

2.0 FUNCTIONAL ANALYSIS

Engineering a large, complex system necessarily involves a decomposition of the system into smaller units. In the case of space missions, it is common and reasonable to impose a hierarchical decomposition, working from larger elements at the top of the hierarchy (Level 1) to smaller elements at the bottom, until the system is sufficiently described to begin implementation (typically Level 5).

The typical system engineering process for JPL missions is functional analysis, also known as functional decomposition. An overview of a generic "Level-N" functional decomposition process is shown in Figure 1.

The process begins by defining the problem to be solved. This step involves developing a Concept of Operations, identifying stakeholder needs, and documenting the boundary between the system being designed and external elements. Next a functional architecture of the system is developed. This includes both a definition of the Concept of Operations in the form of an ordered model of the functions (scenario functions) performed by the system and the intrinsic functionality, with information flow, at Level-N that is required to execute the scenario.

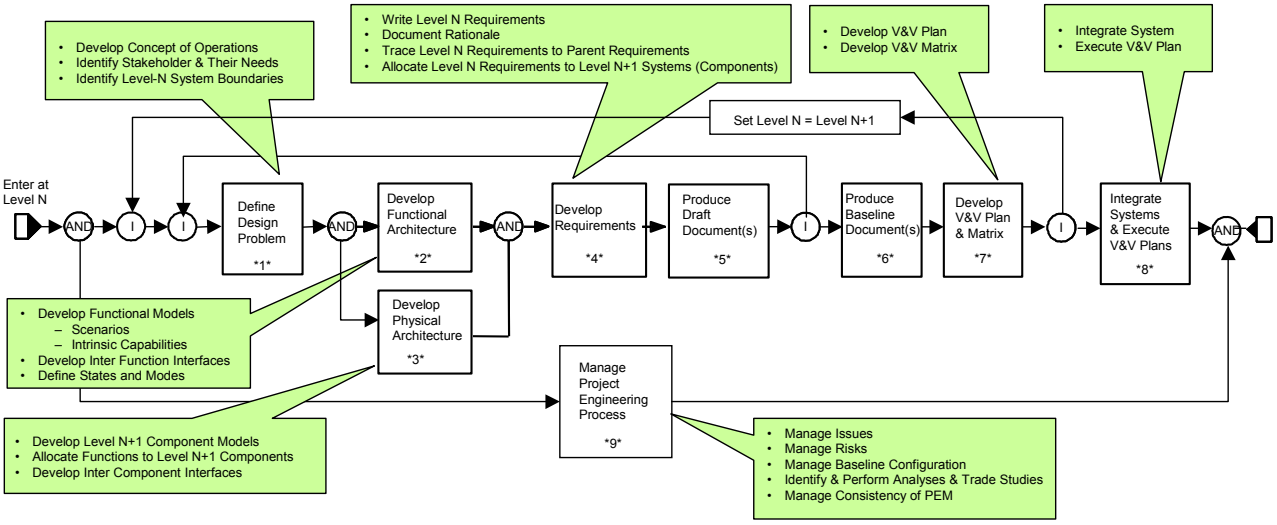


Figure 1 Typical System Engineering Process

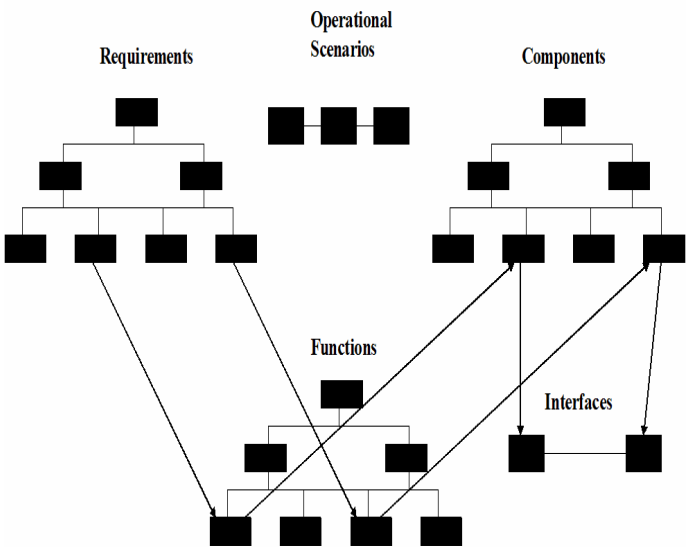


Figure 2 Conceptual Layout of Requirements, Functions and Physical Components

After the functions of the system are defined, the system's physical architecture is defined (with definitions for the product breakdown structure) and system requirements are developed. As functions, physical components and requirements are created; connections between them are created as well so that the relationships linking functions, physical components and the rationale for requirements are also stored.

This iterative process is followed at each level, with design products (requirements, functional models, etc.) at the higher level flowing down to provide a starting point for the next lower level. Ultimately a hierarchy of functions, physical components (product breakdown structure) and requirements are created along with the linkages between the functional, physical and requirement hierarchies. This is shown graphically in Figure 2.

This process is currently performed at JPL using a document-based methodology. The system's functional architecture, physical architecture and requirements are described with documents in natural language. In this approach, while connections within and between hierarchies are defined, they are difficult to verify thoroughly.

The model-based methodology utilizes a model-based language, having a defined syntax (structure) and semantics (meaning), in a computer-aided model-based systems engineering tool. The advantage is that connections within and between hierarchies can be created and machine audited. For example, the user can create a system

engineering tool script to verify that all requirements have been addressed.

In addition, metadata can be stored with each function, requirement and/or physical component. The advantage here is that design data, for example, can be stored with the physical components.

Furthermore, the computer-aided model-based systems engineering tool can produce artifacts that are both machine-readable and human-readable. This means that documents, such as review gate products, can be produced from the repository of system engineering information and that data can be extracted to execute simulation-based verifications. Thus the computer-aided model-based systems engineering tool becomes the central, single repository of system engineering knowledge on the project. While these principles apply to virtually any computer-aided model-based systems engineering tool, the MBED pilot projects used CORE from Vitech [3].

3.0 STATE ANALYSIS

While functional analysis provides a good static view of the system, its ability to represent dynamic behavior is severely lacking. Understanding dynamic interactions typically encountered in the design of complex systems, such as autonomous unmanned air vehicles, robotic vehicles and spacecraft, and automated ground infrastructure, is crucial.

In this formulation phase model-based engineering design process, the dynamic aspects of requirements analysis and design synthesis are addressed with mission planning and performance simulation (described in the following sections). To address dynamic aspects of functional analysis, behavioral modeling is used.

Behavioral modeling identifies the important state variables in the system, capturing the causal effects among these state variables (under both nominal and off-nominal situations), and describing how state variables change under the influence of other state variables and commands issued by a control system.

Behavioral models of this type are invaluable, in that they can be used for multiple purposes, including:

- Informing the design of flight and ground software (e.g., estimation and control algorithms);
- Using them directly in model-based estimation & control software (e.g., Kalman filters);
- Informing the design of fault protection mechanisms (models of nominal and off-nominal behavior can feed into fault tree and FMECA analyses, risk analyses, and fault monitor/response design.
- Feeding directly into simulations, as described in Section 6.0, below; and
- Using them for planning and scheduling purposes (including automated planning and scheduling, either on the ground or onboard the spacecraft).

A novel model-based systems engineering methodology, called *State Analysis* [4], has been developed to complement the functional decomposition approach and better address the complexity challenge. It provides a methodical and rigorous approach for:

- Modeling behavior in terms of system state variables and the relationships between them (*state-based behavioral modeling*);
- Capturing mission objectives in detailed scenarios motivated by operator intent (*goal-directed operations engineering*); and
- Describing the methods by which objectives will be achieved (*state-based software engineering*).

The work described in this paper focuses solely on the state-based behavioral modeling aspect of state analysis, which provides an iterative process for discovering state variables of the system under control and for incrementally constructing the model. The steps in this process are as follows:

1. Identify needs – define the high-level objectives for controlling the system.
2. Identify state variables that capture what needs to be controlled in order to meet the objectives, and define their representation.
3. Define state models for the identified state variables – these may uncover additional state variables that affect the identified state variables.
4. Identify measurements needed to estimate the state variables, and define their representation.
5. Define measurement models for the identified measurements – these may uncover additional state variables.
6. Identify commands needed to control the state variables, and define their representation.
7. Define command models for the identified commands – these may uncover additional state variables.
8. Repeat steps 2-7 on all newly discovered state variables, until all relevant variables and effects are accounted for.
9. Return to step 1, this time to identify additional objectives, and proceed with additional iterations of the process until the scope of the mission has been covered.

This modeling process can be used as part of a broader iterative incremental system and software development process, in which cycles of the modeling process can be interwoven with concurrent cycles of software implementation.

JPL's state analysis approach is novel and unique in three ways:

- It is based on a state-based control architecture (see Figure 3) and leverages a core set of underlying architectural principles [5].

- When coupled with a state-based software architecture [5], State Analysis provides a common vocabulary for systems and software engineers to communicate, and a common set of architectural elements such that the gap between the requirements provided by the systems engineer and the software developed by software engineer can be minimized. More specifically, it defines *direct mappings* from the system requirements (expressed in the form of behavioral models) to software specifications, and from these software specifications to implemented software artifacts.

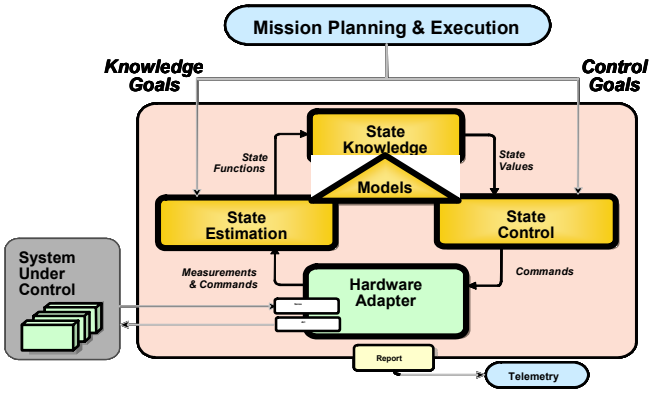


Figure 3 State-Based Control Architecture

- It considers the full breadth of system state variables (e.g., dynamics, environmental states, device status and health, parameters, resources, etc...) and allows for documentation of models using whatever representation is most appropriate (differential equations, state charts, tables, pseudo-code, textual descriptions, etc...).

State analysis produces and compiles information that is traditionally documented in a variety of systems engineering artifacts, including Hardware Functional Requirements, Failure Modes & Effects Analyses, Command Dictionaries, Telemetry Dictionaries and Hardware-Software Interface Control Documents. Rather than separate this information up into disparate artifacts, state analysis captures the same information in a State Analysis Database tool [6] that has been structured to prompt the state analysis process.

The tool design promotes state discovery, and ensures that the models and other requirement artifacts are consistent with the state analysis methodology and state-based architecture. Furthermore, the database schema has been developed to map directly into requirements on adaptations of a state-based control system software framework [5]. In these ways, the database ensures a rigorous project development, from requirements analysis, through software design and implementation, to verification and validation.

4.0 Synthesis of Functional and State Analyses

An important concept in MBSE is that the linkages between and within the functional, physical and requirement hierarchies are explicitly defined and machine auditable. Not only does this allow for an assessment of the completeness and integrity of the model but it allows engineers to assess the impact of changes to the system. With the introduction of state analysis, it is important to synthesize it with functional analysis. As mentioned above, the scope of this effort was limited to integration of the state-based behavioral modeling aspect of state analysis. Integration of the goal-directed operations engineering and state-based software engineering aspects of state analysis are left for future work.

State and functional analysis synthesis was achieved by integrating the State Analysis Database tool schema into a general purpose, computer-aided, model-based systems engineering tool, resulting in a unified model-based systems engineering environment. As previously noted, the MBED pilot project used CORE from Vitech. However the State Analysis Database schema is portable and can be straightforwardly implemented in any systems engineering tool that allows for schema modification.

The synthesis was broken down into 3 subtasks: (1) integration of the state analysis and functional analysis schemas, (2) infusion of the integrated schema into the CORE tool, and (3) demonstration of the integrated approach through modeling of an example system (in this case, the JUNO power subsystem). Subtasks (1) and (2) are described below in greater detail; the results of subtask (3) are described in Section 8.2.1 of this paper.

4.1 Schema Integration

The first sub-task was to integrate the state analysis schema into the functional analysis schema embodied in CORE. This consisted of defining the appropriate set of relationships between elements of the state analysis schema (confining our scope to the state behavior modeling aspect of state analysis) and elements of the functional analysis schema described in Section 2.0, above.

Figure 4(a) shows the relevant elements of the functional analysis schema, along with the relationships between these elements. As shown in Figure 4(b), this schema was augmented with the following state analysis elements: *state variables*, *commands* and *measurements* (*goals* are also shown in the diagram, but, as discussed above, the goal-based operations engineering aspect of state analysis was outside the scope of this task). As shown in the figure, relationships were defined between the functional and state analysis elements, (for example, functions act on *state variables* by producing *commands* that affect them, based on data provided to them through *measurements*).

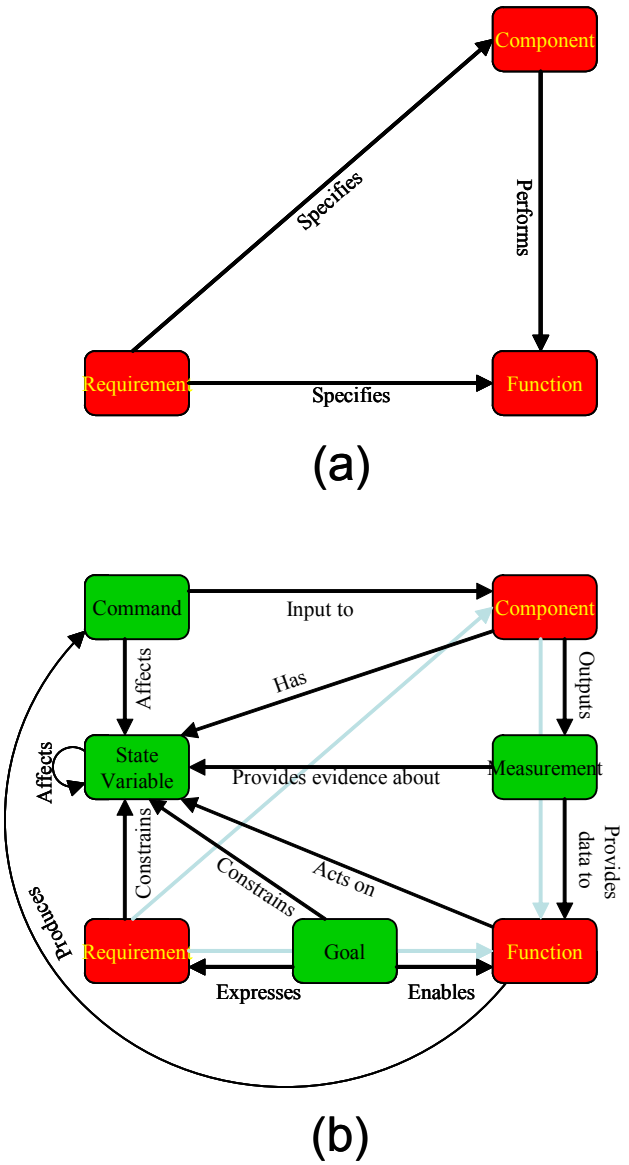


Figure 4. (a) Functional analysis elements and relationships (b) the elements and relationships of state analysis synthesized with functional analysis

4.2 Infusion into the CORE Tool

Given the proposed integration of the two schemas defined in the previous section, the next subtask was to infuse this integrated schema into the selected model-based systems engineering tool CORE. This involved editing the CORE database schema to add the state analysis elements described above as new classes.

For each new class, appropriate attributes were defined, (for example, the state variable class defines fields for description, value representation, and behavior models). Finally, the relationships shown in Figure 4(b) were implemented as relationships in the schema (for example, each state variable can have pointers to other state variables, and to any requirements that constrain it).

The resulting integrated schema enables systems engineers to straightforwardly navigate between the artifacts resulting from their functional and state analyses. Furthermore, it provides a mechanism for automatically auditing their models for satisfaction of certain completeness criteria, (for example, each state variable affects or is affected by at least one other state variable; each component has at least one state variable; each measurement provides data to at least one function; etc).

5.0 Mission Planning

The mission planning component of the MBED process develops the Concept of Operations and the mission scenarios that are used to drive the models of the spacecraft subsystems in the spacecraft performance verification component. Mission scenarios are descriptions of "what happens in space" to accomplish the objectives of the mission and are often described using timelines and time-ordered listings of events.

The mission planner is a system engineer with a focus on developing the understanding of how the mission events will proceed from launch until the end of the mission. The mission planner develops descriptions of how the payload will be used to accomplish the scientific objectives; how the project systems support and constrain the science observations; how the mission design (including launch strategy and orbit) defines the timing and geometry for the observations; and how the operational strategy (including experience from past missions) is applied to develop the order and specifications of the mission events. For flight projects, this overall description of the mission is documented in the Mission Plan. The overall description is built up from scenarios that describe specific periods of the mission (for example, launch, commissioning of the spacecraft, maneuvers, routine and specialized science observation periods, data playbacks, etc.).

For MBED, the mission planning process begins by reading the mission's Phase-A concept study report. This document provides a fairly detailed mission design, operational plans, science and mission objectives, technical implementation, and project schedule. Next, the mission principal investigator is interviewed to further understand more specifics about the instrument operation and science strategy. Questions are then directed to the spacecraft manufacturer (in-house or contractor) to better understand the subsystem capabilities and constraints (for example, turn rates, antenna patterns, etc.). This part of the process includes participation by engineers building the subsystem models. A trajectory simulation is then built using the SOAP tool (Satellite Orbit Analysis Program, created by Aerospace Corporation) [7]. to understand the timing of

geometric events, including tracking station contacts, and the spacecraft geometry relative to pertinent celestial bodies (Sun, Earth, target planet/asteroid/comet), including pointing of its solar arrays and communications antennas. Instrument pointing constraints are also investigated in the SOAP simulation to determine times during the mission when potential hazards/obstructions may occur.

With this understanding of the science objectives, instrument operations and constraints, spacecraft subsystem characteristics, and the trajectory geometry, mission scenarios are identified to demonstrate typical and stressing cases for operating the payload and spacecraft during the course of the mission. When these scenarios have been detailed with all of the relevant subsystem activities, they are converted to activity plans that can be utilized by the subsystem modeling tool(s) in a simulation run (the format of these activity plans is negotiated between the mission planner and the engineers building the subsystem models). The results of the simulation run are compared with the requirements that are built into the system engineering model to verify compliance.

In addition to verification of the existing requirements, simulation runs of these scenarios may also bring new requirements to light. Requirement discovery from examining scenarios is a standard part of the iterative process of mission design and requirements development.

6.0 S/C PERFORMANCE SIMULATION

In a deep space mission's Preliminary Design Review, the project team seeks to demonstrate to a review board that they understand the mission objectives and that they are capable of achieving them. One aspect of this is showing that the conceptual S/C design satisfies the mission's requirements. The document-based approach for doing this is to create design documents and have the review board evaluate them. The model-based approach is to simulate the mission and check the numerical results against the technical requirements.

The model-based performance requirements verification process requires two pieces of software: an integrated suite of S/C performance and resource simulation models and a requirements checker that compares the time-ordered results to the mission's technical requirements. The process is then straightforward:

1. Configure with design data from the systems engineering model and run the integrated simulation models with every available mission plan.

2. Compare the mission's technical requirements to all simulation results and report any violations.

While these principles apply to virtually any spacecraft performance and resource simulation models, the MBED pilot used JPL's Integrated Spacecraft Analysis (ISCA) suite. Data transfer between the computer-aided model-based systems engineering tool, CORE, the mission planning process and simulations was coordinated using JPL's Inter-application Communication Executive for Computational Analysis Tools (ICECAT) described in Section 6.2.

6.1 INTEGRATED SPACECRAFT ANALYSIS

The simulation models used on this task were the multi-mission models (MM Models) [8], integrated for an Integrated Spacecraft Analysis (ISCA).

ISCA is designing, developing and/or acquiring dynamic, multi-mission, multi-use, multi-scale (variable fidelity) simulation models and an integrating framework that can be used cross-mission and throughout the mission lifecycle to predict spacecraft performance and resources. The goal is to provide ready-to-use, dynamic S/C performance and resource prediction capabilities in a single integrated package.

The simulation models consist of finite state models, physics equations, algorithms and/or heuristics needed to make acceptable performance and resource predicts. Capabilities will include the ability to predict and plot spacecraft performance and resources in the following areas:

- Power
- Thermal
- Propulsion
- Attitude Control System (ACS)
- Command and Data Handling (C&DH)
- Telecommunications (flight & ground)
- Orbital mechanics & S/C trajectory (SPICE)

Structural and environmental models are integrated as well but they do not provide S/C predicts. These capabilities are integrated using a Multi-Mission Simulation Framework (MMSF). Figure 5 shows a conceptual design of MMSF.

MMSF consists of four main modules: a Simulation Manager, a Data Manager, Model Interface and a Multi-Mission (MM) Model Interface. Internally the Simulation Manager coordinates simulation time and events for all models. The Data Manager handles data sharing between models as well as error and results reporting. This provides a tight model coupling so that when state changes occur in one model, such as solar array power, it immediately affects other models.

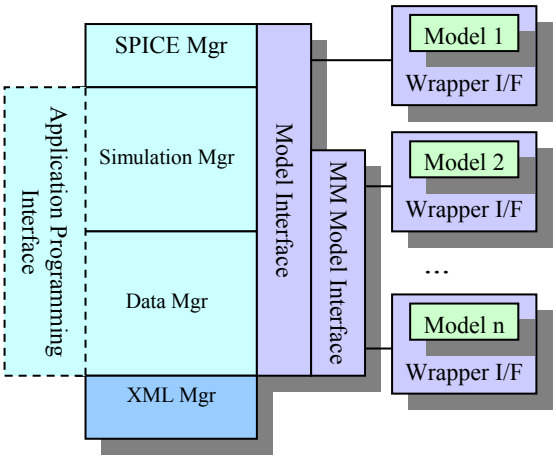


Figure 5 ISCA Multi-Mission Simulation Framework

Access to these services is provided by the Application Programming Interface (API) and Model interfaces. The MMSF API (whose calls are incorporated in the Simulation and Data Managers) provides the necessary functions for 3rd party software (driver module) to run an ISCA. It allows driver modules to control the execution of the various Simulation Engine phases (configure, initialize, advance, finalize) and permits access to the Data Manager for retrieving/modifying data from any model during a session.

Simulation models are integrated into ISCA through the MMSF Model and Multi-Mission Model Interface. These interfaces allow developers to easily plug "wrapped" simulation model components into the framework and get them to operate with other components. Simulation model components can be from any source and be of any type, through they typically correspond to some aspect of a physical S/C assembly such as a data bus throughput model. The baseline set of MDAS Multi-Mission models use the optimized MM Model Interface while others use the more generic Model Interface.

ISCA results are not scripted. Instead the simulation models will change state in response to inputs from the user. The framework coordinates model responses so that they can have an immediate effect on other models integrated into the
simulation. To perform an analysis using ISCA the user must provide two types of information: the system configuration and the mission timeline. This information can be stored in files and read into the system before the simulation is executed or provided through the API as the simulation is executing.

Configuration files are stamped with a particular mission time and contain parameters for each spacecraft in a system such as: design, environment and states, including software states, switch states, temperature, battery state of charge, attitude, location, ambient temperatures, atmospheric conditions, as well as any telecom relay spacecraft, and ground stations. There are two types of configuration files, the Baseline Configuration file and Situational Configuration file.

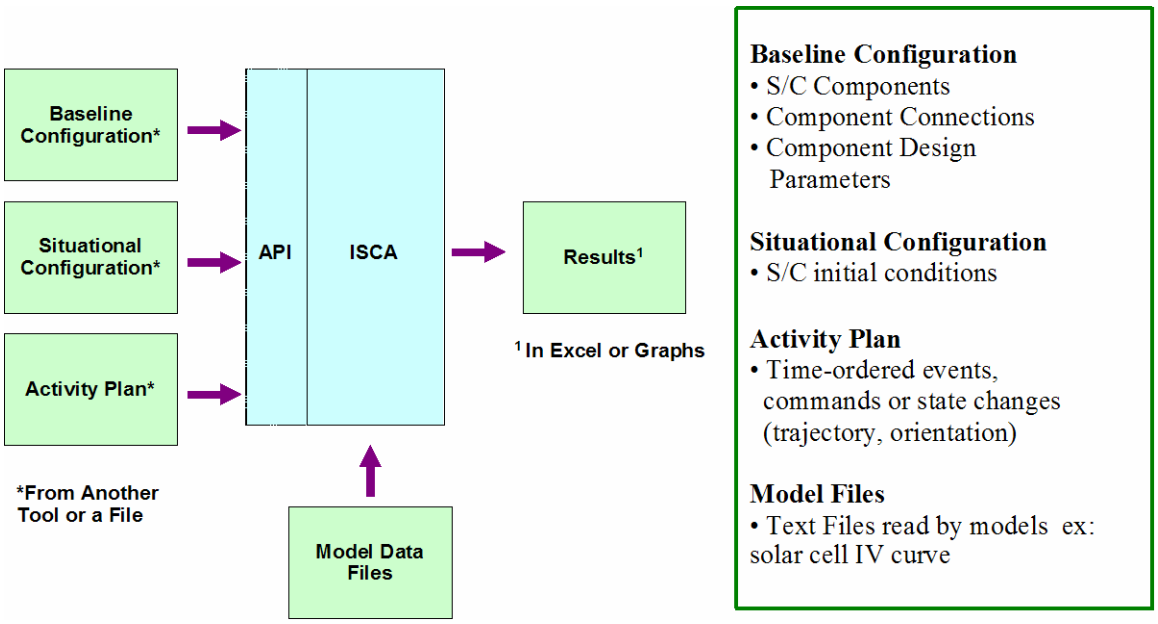


Figure 6 ISCA Input and Output Files

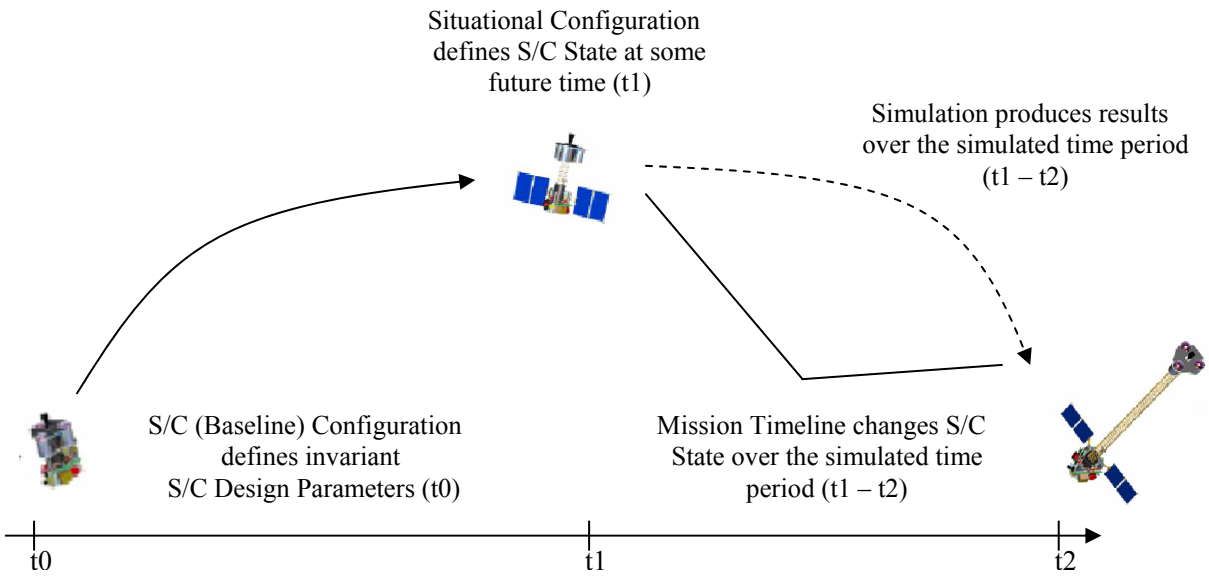


Figure 7 ISCA Input and Output Files used in Simulation

The Baseline Configuration file represents the configuration of each spacecraft at the beginning of the mission. There is typically a single baseline configuration file for each mission. Its purpose is to provide a basis for all configurations used throughout the mission. As such, spacecraft design parameters are important, whereas state and environmental parameters are not since they are expected to be contained in Situational Configuration files.

The Situational Configuration files act as an overlay to the Baseline Configuration file and are used to contain system state and environmental parameters for a particular point in time. Design parameters are only put in this file if a hardware fault has occurred.

Timeline files (also known as Mission Scenarios or Activity Plans) or are time-ordered lists of commands or parameter value changes. When the simulations are run, these values are sent to the simulation models at the specified time.

6.2 Inter-application Communication

One of the many challenges in MBED FY06 was the integration of the Mission Planning, System Engineering, and S/C Performance Analysis processes. On many flight projects there are standalone tools running on various platforms and producing data that cannot simply be input into other tools. One of the model-based engineering goals is to allow seamless transfer of data between a heterogeneous set of distributed tools. In MBED FY06 this was achieved by adapting a COTS framework called Eclipse to create an Inter-application Communication Executive for Computational Analysis Tools (ICECAT) application.

Eclipse is an open source community whose projects are focused on providing a vendor-neutral open development platform and application frameworks for building software [9]. The Eclipse Foundation is a not-for-profit corporation formed to advance the creation, evolution, promotion, and support of the Eclipse Platform and to cultivate both an open source community and an ecosystem of complementary products, capabilities, and services.

Eclipse has formed an independent open eco-system around royalty-free technology and a universal platform for tools integration. Eclipse-based tools give developers freedom of choice in a multi-language, multi-platform, multi-vendor environment. Eclipse provides a plug-in based framework that makes it easier to create, integrate and utilize software tools, saving time and money. By collaborating and exploiting core integration technology, tool producers can leverage platform reuse and concentrate on core competencies to create new development technology. The Eclipse Platform is written in the Java language and comes with extensive plug-in construction toolkits and examples. It has already been deployed on a range of development workstations including Linux, HP-UX, AIX, Solaris, QNX, Mac OS X and Windows based systems.

Since each tool has its own input/output data requirements and since the tools used within each domain vary, with new tools being developed or purchased, an interface plug-in module was created for each application used in MBED. ICECAT acts as the executive of the system. It is responsible for retrieving input data such as the requirements and S/C design, mission planning products (activity plans), executing integrated tools sets such as the ISCA simulation models and requirements verification software, and managing the storage for any end products generated (performance analysis results, logs, etc.). In addition, the framework is the user's interface into the entire system. It is responsible for processing user requests to access data, run specific tools, and store various results. It

can also display data and assist users during a session through a set of wizards.

The core framework software module communicates with the distributed set of tools by issuing command to a set of interface modules, which in turn access the necessary tools (see Figure 8). Each of these modules is dedicated to a specific tool that needs to exchange data. Each module contains 2 basic components – data processing and communication protocol. The data processing component is responsible for:

- Accepting ICECAT Core Software data as input
- Mapping this data into the appropriate input data for the dedicated tool.
- Formatting the data before sending it to a specific tool
- Parsing this result data and
- Mapping and formatting the result data into ICECAT Core Software data.

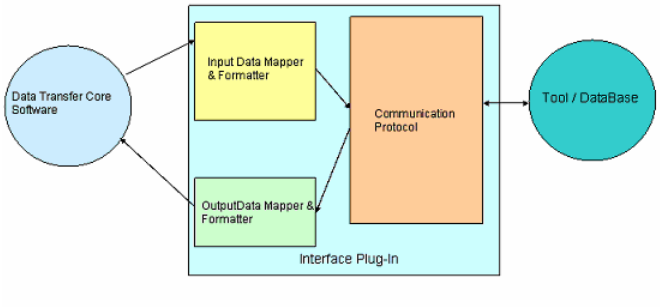


Figure 8 Interface Plug-In Architecture

The communication component is responsible for:

- Invoking the tool and passing the formatted data
- Accepting tool result data

The set of tools that will be part of the system may change as new software is purchased and/or developed. ICECAT's plug-in architecture design allows these new tools to be added without impacting the existing set of tools. When a new tool is to be added, an interface plug-in module must be created for that tool. Since the knowledge of the new tool and its data requirements are captured in the plug-in, the core data transfer software and the existing plug-ins are not affected.

This approach to adding new tools allows for the functionality to be extended while reusing all of the existing software elements.

7.0 NuSTAR Pilot

MBED FY06 applied integrated, model-based system engineering, mission planning and performance requirements verification techniques to the NuSTAR project. Since this was a research task these capabilities were demonstrated by "shadowing" the flight project. Shadowing in this context means running the task as a flight project but not providing any deliverables to the flight project.

Unfortunately the NuSTAR project was canceled two-thirds of the way through its formulation phase. Nevertheless many of the goals of the pilot were achieved as will be described in the following sections.

7.1 NuSTAR Mission Description

The NuSTAR Mission is a Small Explorers (SMEX) mission for observing and imaging X-ray sources in space. This mission uses a technology that is new to space applications: a focusing hard X-ray telescope operating in the energy range from 6 to 80 keV. The design of this telescope eliminates high detector backgrounds and allows for true imaging of these hard X-ray sources.

While the mission was selected for a Phase A study in 2003 and further study in 2005, it was ultimately cancelled by NASA in 2006 due to agency budget limitations. Before cancellation, the project was targeting a November 2007 launch.

The primary objectives of this mission are:

- To conduct a census of black holes on all scales by performing deep wide-field surveys of the extragalactic field and the Galactic center.
- To map radioactive material in young supernovae remnants, to study the birth of the elements and to understand how stars explode.
- To detect relativistic jets of particles from extremely active galaxies, to understand what powers giant cosmic accelerators

To achieve the sensitivities required for these objectives, there is an array of three co-aligned hard X-ray telescopes, requiring a 10-m focal length. This focal length is provided by a 10-m mast that is deployable on-orbit.

The proposed NuSTAR spacecraft was to be built by General Dynamics/Spectrum Astro. It would have provided 3-axis pointing control of the instruments using reaction wheels and a star tracker. Energy would have been provided by two solar arrays with a single gimbal axis to allow pointing to the sun in any attitude and a battery to provide energy during the solar eclipse on each orbit. The telecommunications design would have provided communications at S-band using two hemispheric patch antennas, with the capability to return science data to a 10-meter ground antenna at a rate of 4.12 Mbps. The standard science collection rate from the X-ray telescope would have been 15 kbps, with the spacecraft providing data storage of

16 Gbits.

The NuSTAR spacecraft was to be launched into a 525-km circular, near-equatorial orbit, with an inclination of just 1º, minimizes radiation effects on the instrument data. Upon arrival in this orbit, the observatory would have deployed its solar arrays, acquire signal from the primary ground station, and performed a checkout of all systems. At fifteen days after launch, the observatory was to deploy the instrument mast. It would then have performed an alignment to verify that the optics are focused on the detectors and calibration procedures to confirm operation of the instrument as a whole. After confirming the operation of the telescope, the observatory would have begun its science operations.

In its science operations phase, the observatory will perform surveys and pointed observations. To perform a survey, the observatory will sweep the telescope's boresight across a pre-defined portion of the sky, gathering data from any X-ray sources residing in that area. For pointed observations, the observatory would point the telescope directly at a specific target, collecting enough data to produce a high-resolution X-ray image of the target.

NuSTAR requires a daily downlink to return the science data collected from its current target. The primary ground station for support of this mission is Malindi (Kenya), with a backup station of Kourou (French Guiana). With the near-equatorial orbit, NuSTAR has visibility of both stations on every orbit. Because of the rotation of the Earth, the spacecraft's antenna configuration, and its current orientation, some orbits have greater potential for telecom passes than others.

7.2 NuSTAR Requirements and Design

NuSTAR functional behavior, physical architecture and Level 2 requirements were captured in CORE Workstation 5.1 following the procedure outlined in Section 2.0. CORE was selected because of its ability to document the critical systems engineering elements (e.g., requirements, functions, components, interfaces, risks, issues) and could be adapted to specific elements required by the team.

The following functions, physical components, and requirements were entered into CORE:

- **182 NuSTAR functions:** The basic function was "Gathering Science Data" and that was decomposed to functions that would satisfy the NuSTAR requirements and can be allocated to a specific component.
- **330 NuSTAR physical components:** These physical components included common spacecraft items like: solar array panel, camera, CPU, or battery. Design information was entered into the property sheet for each component entered. Each component type had a unique template that described design parameters typically associated with that type of component. Templates were created using the CORE Schema Extender.
- **194 NuSTAR requirements:** Since the NuSTAR requirements were already available in Word documents, the MBED team used the CORE Requirement Extractor tool to copy them into the CORE database.

Requirement verification tools need easy access to the numerical information within the requirements. The Schema Extender tool was used to modify the basic CORE requirement template to include these fields. Since most measurable requirements are in the form:

Measurement <>= Value in Units for some Duration

Fields were added for these numerical values. Like the intrinsic functions and physical architecture, the requirements were linked together in CORE in a hierarchical manner. CORE scripts were then used to audit the overall model for completeness and integrity.

Three gate products were created by the MBED NuSTAR pilot from the CORE system engineering model:

- NuSTAR Level 2 Requirements Document
- NuSTAR Significant Risk List
- NuSTAR Mission Scenarios

The NuStar Level 2 Requirements document and the NuStar Mission Scenarios document were created using report scripts which were obtained from Vitech Corporation. The report scripts were run from within CORE, but before they could be used, the schema was modified in the CORE database to support the reports.

Most of the system engineering data for NuStar was already resident within the CORE database, but additional report-specific data was entered into the database as well as links to graphics. Each section of the report was then configured by linking the correct report script with the appropriate data within the CORE database. Some of the report scripts generated CORE graphics. To include these in the report the graphics required some size adjustments so they could be optimally viewed.

The risk report was created by a different mechanism since the Vitech report scripts did not support some of the field types. The CORE database was dumped into an eXtensible Markup Language (XML) file. A custom XML parser was then written (in perl), which extracted out the risk elements from the XML file, and then stored in a new XML file. The new XML file was also reformatted into a simpler XML format to aid in the report processing. A tool called JasperReport running under the Eclipse environment was then used to layout and create the report using the new

XML file as an input.

7.3 NuSTAR Mission Planning

The baseline plan for the mission is to observe a series of targets that have been organized into a schedule. Depending on whether the target is to be surveyed or observed, the observatory will sweep the telescope across or stare at the target. Pointed observations have durations that range from a week to a month, but survey can take 1-2.5 months to complete.

Scheduling of activities for a scenario is based on the expected behavior of the spacecraft and its instrument(s). The behavior of the spacecraft is defined by the orbit in which it resides and the attitude control system it uses. For determining geometric events (eclipses, occultation, visibility), the orbit must be understood. SOAP is a very powerful tool in gaining an understanding of an orbit; it quickly presents a visual representation of an orbit based on the elements the user enters, and provides an analysis environment that can quickly determine the geometric events. Figure 9 shows a representation of the initial NuSTAR orbit.

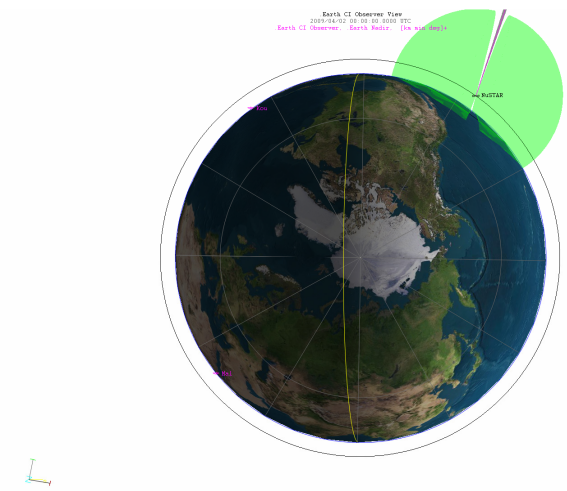


Figure 9 NuSTAR Initial Orbit – Polar View

Also shown in this figure are a representation of the spacecraft antennas' fields of view (FOV) and the telescopes FOV. To represent the FOVs of the observatory components, an understanding of the attitude control concept is required. In the case of NuSTAR, the spacecraft maintains a viewing orientation with the telescope boresight pointed at the science target and the solar array gimbal axis pointed normal to the Sun line; the solar arrays have a one-axis gimbal and can maintain full-Sun on the arrays in this orientation.

One of the main concerns for any mission is the capability to communicate with the ground. Using the analyses available in SOAP and the expected performance of the spacecraft antennas and ground stations, occurrences of potential telecom contact can be determined. To determine the occurrences of telecom contact, the antennas for the spacecraft and the ground station are modeled in SOAP. For the ground station, the antenna configuration is simple: a nadir-facing sensor with a half-cone angle of 85˚. This configuration represents the antenna's capability to track the spacecraft as it passes the station, assuming a 5˚ elevation mask. The spacecraft has 2 antennas facing in opposite directions. The antennas are omni-directional with boresights pointed in directions normal to the telescope boresight and the solar array gimbal axis; this configuration means that the antennas are pointed roughly toward and away from the Sun. Because of antenna performance characteristics and obstruction by some parts of the spacecraft, each antenna falls short of covering a full hemisphere; the half-cone angle for each spacecraft antenna is 85˚, leaving a small gap in coverage around the plane normal to the antennas' boresights. This gap can be seen in Figure 9, where the telescope is pointed toward the upper right of the figure and the Sun is directly right. The spacecraft switches between these antennas as required by the contact schedule. SOAP analyses provided listings of all potential telecom contacts during the operational periods of these scenarios. In most cases, a single ground station pass consisted of two contacts (one for each antenna on the spacecraft) with a short gap between them; there are 2 passes each day that are single-contact passes, one sunlit and one during eclipse.

A set of five scenarios were constructed to test a variety of mission requirements. Each scenario represents either a typical case or a case that stresses some aspect of performance. The first two scenarios present typical performance of the observatory at the beginning of the mission. The difference between these scenarios is the treatment of ground station contacts; the first uses single-contact passes that occur during eclipse periods (stressing power usage), and the second uses an expected data delivery time for scheduling the contact passes without the restriction to single-contact passes. These two scenarios could be used as the beginning of a trade study on downlink operations. The third scenario also presents typical operations, but uses a different science target and occurs in the later part of the mission; the observatory does not have orbit maintenance capability and its orbit decays over time, resulting in a shorter orbit period and shorter contact pass durations (stressing data return requirements). The fourth scenario presents another case of typical operation in the earlier part of the mission, but it uses a different science target, a different delivery time, and restricts passes to sunlit single-contact passes; this case would provide more data for use in the trade study. The final scenario demonstrates the execution of a worst-case target change associated with acquiring a target of opportunity; the spacecraft reorients by performing 3 single-axis rotations to maintain Sun-pointing with the solar arrays and to make use of the most stable configuration for most of the retargeting procedure. This

group of scenarios addresses most of the operational modes and procedures that would be utilized over the life of the mission.

This set of scenarios was constructed into a set of activity plans for use in the performance simulations. The format was negotiated between the mission planner and the simulation software developers.

7.4 NuSTAR Simulation Results

The integrated multi-mission (MM) models were simulated each of the five scenarios NuSTAR design parameters. The results of the MM models were then compared to mission and system requirements to see if the design could meet them. Figures 10, 11 and 12 graphically depict some of the ISCA model predictions versus requirements for a portion of one of the typical scenarios described in section 7.3.

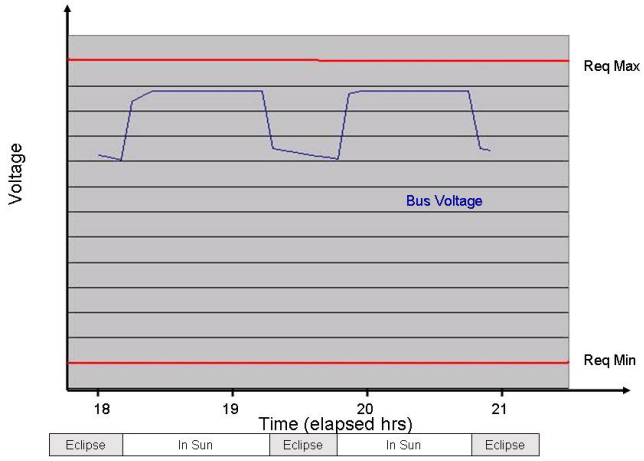


Figure 10 – Bus Voltage vs. Time

Figure 10 shows a graph of the predicted bus voltage as a function of time. The horizontal red lines indicate the required minimum and required maximum allowable bus voltages. The timeline below the plot indicate the periods of time when the NuSTAR spacecraft was alternately in the sun and eclipsed by the Earth.

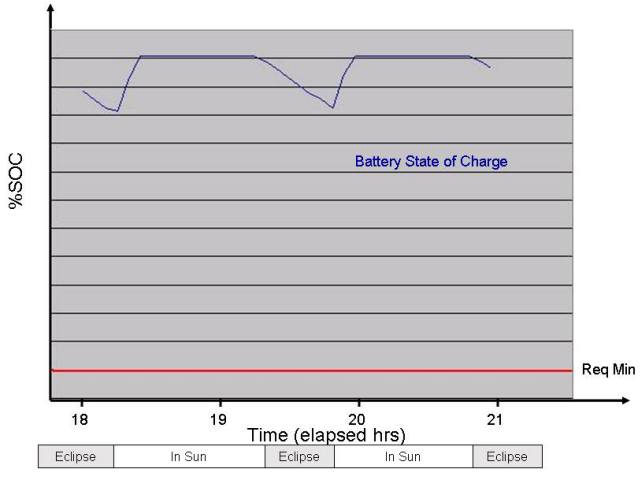


Figure 11 – Battery State of Charge vs. Time

Figure 11 shows a graph of the predicted battery state of charge (SOC) as a function of time. The horizontal red line indicates the minimum state of charge percentage that is allowed by the mission and system requirements. The timeline is the same as in Figure 10.

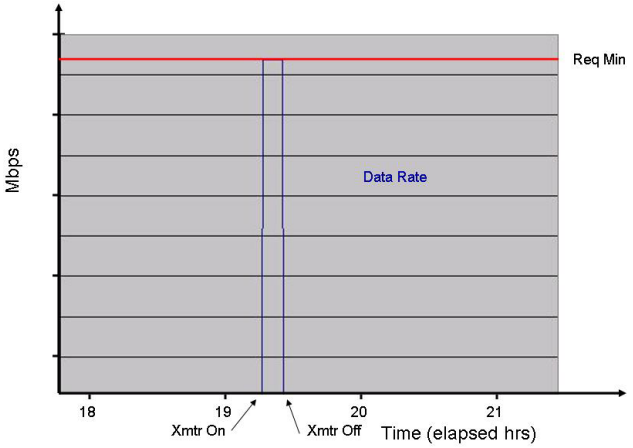


Figure 12 – Maximum Data Rate vs. Time

Figure 12 depicts a graph of the predicted maximum telecommunications data rate as a function of time. The horizontal red line indicates the required minimum data rate for when the NuSTAR spacecraft is downlinking data to the Earth, whether it is the ground station at Malindi or Kourou. The timeline indicates when the scenario commanded the ISCA telecommunications model to perform the link analysis. The "Xmtr On" command from the scenario instructs the models to begin the link analysis, the "Xmtr Off" command instructs the models to end it.

8.0 Juno Pilot

MBED FY06 applied integrated, model-based system engineering, mission planning and performance requirements verification techniques to the Juno project. Since this was a research task these capabilities were demonstrated by "shadowing" the flight project. Shadowing in this context means running the task as a flight project but not providing any deliverables to the flight project.

While still early in the design process, a principal design risk was identified that the power margins may not be adequate in specific mission scenarios given solar cell degradation from the high radiation environment at Jupiter. Therefore, the MBED team applied the principles of model-based engineering design to review the mission requirements, develop a detailed mission scenario, and analyze the power budget using specialized design tools.

8.1 JUNO MISSION DESCRIPTION

In 2005, NASA selected the Juno mission to conduct an in-depth study of Jupiter, the most massive planet in the solar system. Juno will peer through the clouds of Jupiter's atmosphere to reveal the fundamental processes of the formation and early evolution of the gas giant planet, allowing a better understanding of the solar system.

The Juno science investigation will focus on four themes:

- **Origin:** Juno will measure global oxygen and nitrogen by mapping the gravitational field and using microwave observations of water and ammonia.
- **Interior:** Using maps of Jupiter's gravitational and magnetic fields, Juno will reveal the interior structure.
- **Atmosphere:** By mapping variations in composition, temperature, opacity and dynamics, Juno will determine the structure and dynamics of the atmosphere.
- **Magnetosphere:** Juno will measure the distribution of Jupiter's aurora's charged particles, their associated fields, and the concurrent UV emissions of the planet's polar magnetosphere.

To accomplish these objectives, Juno will carry a scientific payload that includes seven instruments: (1) dual frequency gravity/radio science system, (2) six wavelength microwave radiometer for atmospheric sounding and composition, (3) dual-technique magnetometer, (4) plasma and energetic particle detector, (5) radio/plasma wave experiment, (6) ultraviolet imager/spectrometer, auroral distributions experiment, and (7) a color camera.

The mission will be launched in 2011 on an Atlas V launch vehicle, and it will arrive in 2016 via an Earth gravity assist. The science orbit will consist of 30 "11 day" science orbits at a 90-deg inclination to reduce radiation exposure. This orbit will sample a full range of latitudes and longitudes, combining in situ and remote sensing observations.

The Juno spacecraft is spin-stabilized and solar powered, incorporating dual-redundancy to reduce risk. It has three conventional solar array wings, a dual-mode propulsion system, and a mechanically-quiet environment to minimize science disturbances. Overall, the spacecraft design maintains exceedingly high margins with solar power being an exception in specific scenarios, where radiation may degrade the capability of the solar cells.

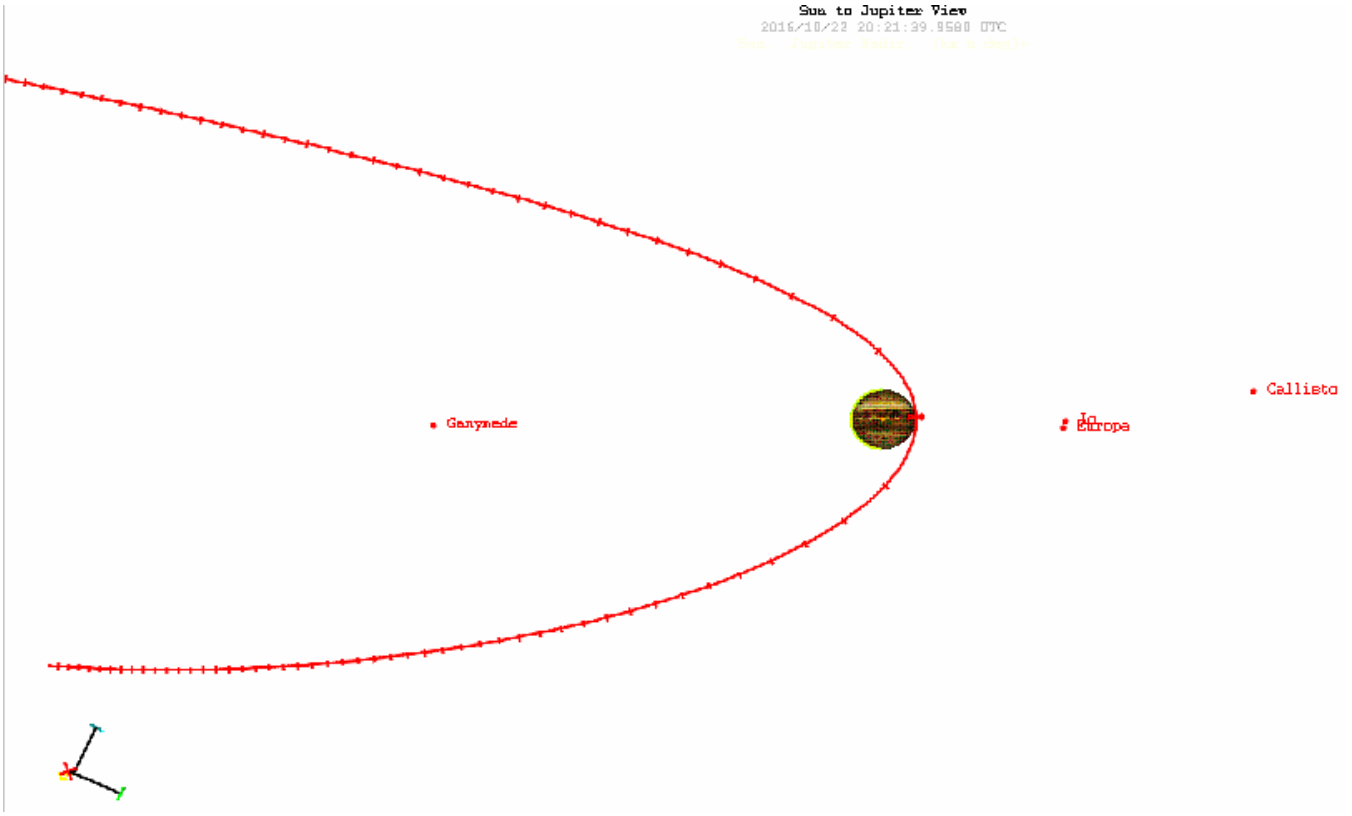


Figure 13 Juno in its expected JOI orbit around Jupiter

8.2 JUNO REQUIREMENTS AND DESIGN

The focus of the Juno effort was on power analysis, particularly during Jupiter Orbit Insertion (JOI). It turns out that JOI was the peak power usage for the entire mission. Short power peaks during the trim maneuvers for each science orbit were also identified.

Requirements, components and functions were entered into the CORE database for analysis. The data was obtained from the Juno Level 2 Requirements document, the Juno Concept Study Report, the Juno Power Worksheet and from discussions with the Juno Engineers. The schema used for the power-related components was directly inherited from schema used previously for NuStar – it was just a matter of entering the Juno data for the batteries and solar arrays. The solar arrays schema was broken into considerable detail including subcomponents for segments, panels, strings and cells. Over 170 components were defined along with over 400 related power modes.

Over 260 requirements were entered into the database, with about 50% of them being power-related. The power margin requirements for Juno were mostly derived. Power margins were broken down by subsystem and mission phase. The total power margin consisted of the Current Best Estimate (CBE) + Contingency + a fixed 13% margin. The Contingency varied by mission phase, and ranged from 11% to 30%, averaging about 20%. Power margins were derived for JOI and the Gravity Science orbit. The derived power margins were then used in the simulation runs. After the requirements, components and function were all linked together in the CORE database, audits were performed to verity the linkages.

8.2.1 JUNO STATE ANALYSIS

In order to validate the integrated Functional and State Analysis schema described in Section 4.0, and to demonstrate the complementary nature of these two model-based systems engineering methodologies, a limited state discovery and modeling process was applied to the Juno power subsystem described in Section 8.1. The resulting State Analysis artifacts were captured in the augmented CORE tool. Figure 14 is an example State Effects Diagram showing the key state variables, commands and

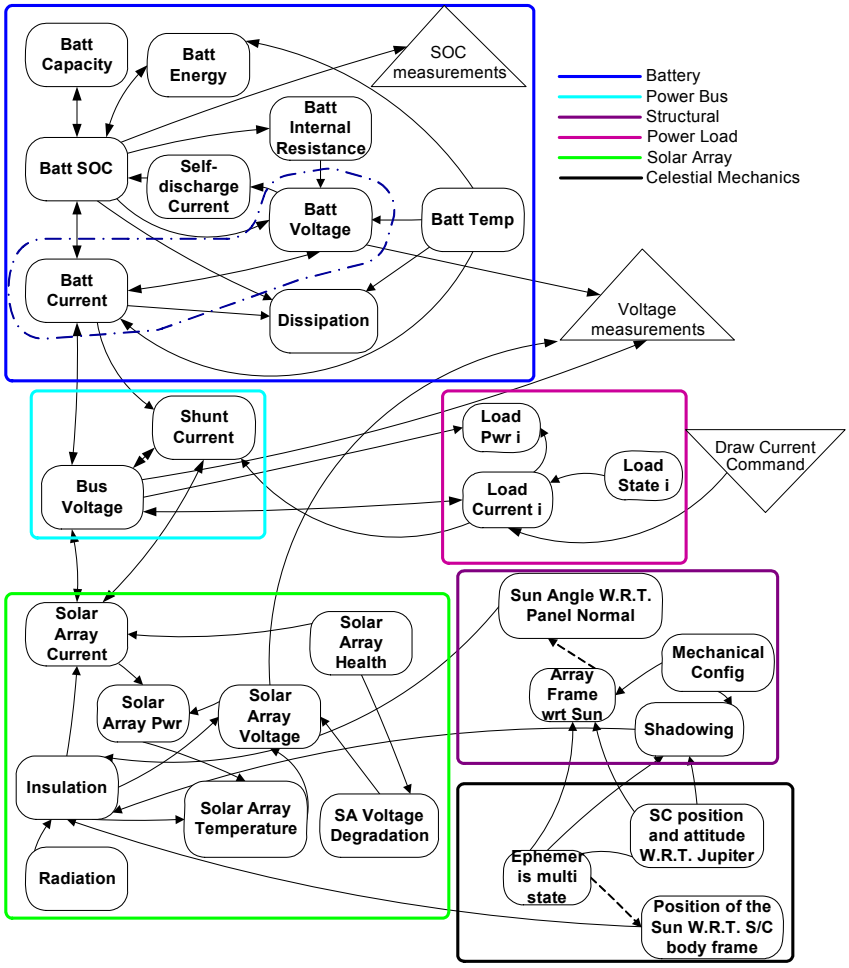


Figure 14 Sample State Effects Diagram

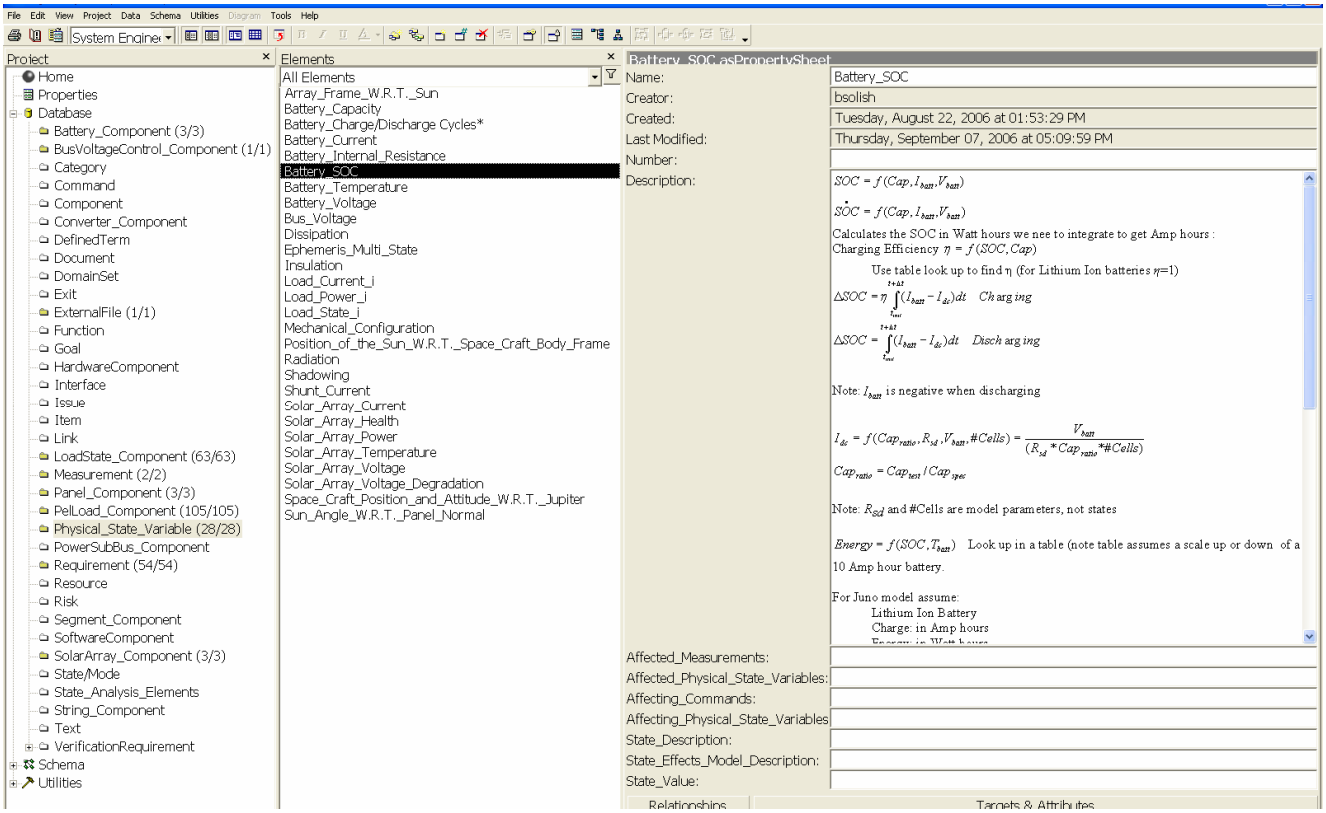


Figure 15 State Variables and associated models are captured in CORE

measurements that were identified for the subsystem, and the causal effects among them.

Populating the integrated database was relatively simple. The first step was to enter each state variable, command, and measurement element under their respective Classes in CORE. This was no different than for any other CORE element. Once the elements were instantiated in CORE, they could be further populated with their attribute information, e.g., state variables must specify how their values are to be represented (for instance, discrete states like Device Operational Modes may use enumerated sets of values {OFF, ON, INITIALIZING, etc.}, while continuous states like Device Power Consumption may be represented as floating point numbers in units of Watts). The final step was to "link" the various elements with each other, by instantiating the relevant relationships from those shown in Figure 4(b). Figure 15 shows a screen capture from the CORE tool, showing the list of 28 state variables that were defined for this subsystem, and showing the behavior modeling details associated with one of these state variables, the battery state of charge.

8.3 JUNO MISSION PLANNING

Because of the focus on the power subsystem and the desire for rapid turnaround, the initial mission planning activities focused on identifying the operational scenario that put the most stress on the power subsystem. From examination of the power states and baseline operational scenarios provided by the Juno team, it was determined that the stressing case was Jupiter Orbit Injection (JOI); during the main-engine burn for JOI, the spacecraft places the highest demand on the power subsystem, and the solar arrays are facing away from the Sun, producing no/minimal power. Juno's baseline scenario for JOI was completely developed, but the mission's launch date changed since the scenario's development. Using this baseline, the mission planner developed a new scenario for the new epoch of JOI; the activities are the same as those in the baseline, but the scheduling of the activities was modified to align with the new arrival epoch.

Scheduling of activities for a scenario is based on the expected behavior of the spacecraft and its instrument(s). The behavior of the spacecraft is defined by the trajectory it traverses and the attitude control system it uses. For determining geometric events (eclipses, occultation, visibility), the trajectory must be understood. SOAP is a very powerful tool in gaining an understanding of a trajectory; it quickly presents a visual representation of a trajectory based on the elements the user enters or an ephemeris of the trajectory, and provides an analysis environment that can quickly determine the geometric

events.

The trajectory for Juno for the selected epoch was provided by the Juno team and was analyzed for the relevant geometries. The geometries of interest are the line-of sight between the ground stations and Juno, the potential for eclipses/occultations, and the solar array attitude relative to the Sun. Although the telecom subsystem is not being modeled, the ground station contacts are relevant to the scheduling of activities, including transitions by the transmitter between standby and transmit.

For the eclipse/occultation concerns, the trajectory was designed to not experience these types of events from Jupiter approach through end of mission (EOM); the SOAP analysis confirms this condition. For the solar array attitude, the solar arrays are body-fixed and normal to the spin axis of the spacecraft; the spin axis of the craft is pointed at Earth for a majority of the JOI scenario, but when preparing for and executing the main-engine burn, the spacecraft precesses from Earth-point to a fixed attitude for burn execution, an attitude which points the spin axis approximately normal to the Sunline.

The SOAP analysis produced tabular output of the ground station contacts and the profile of the solar array attitude. These data facilitate scheduling and construction of the activities plan for the JOI scenario.

8.4 JUNO SIMULATION RESULTS

The integrated multi-mission (MM) models simulated the JOI scenario using Juno design parameters for both current best estimate (CBE) and CBE plus contingency. The results of the MM models were then compared to mission and system requirements to see if the design could meet. Figures 16, 17 and 18 graphically depict some MM model predictions for the JOI scenario. The predictions are for the current best estimate design parameters.

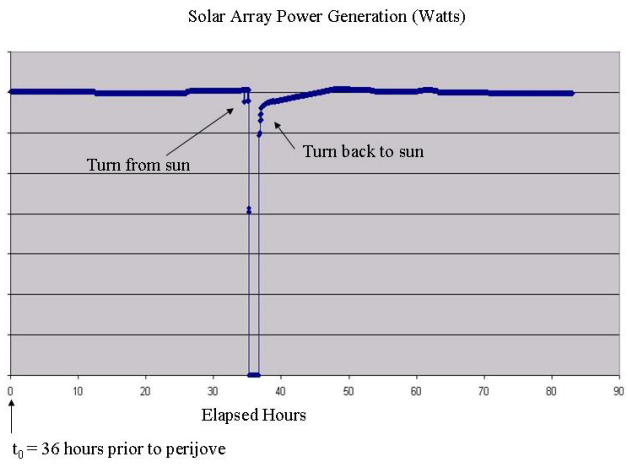


Figure 16 – Solar Array Power Generation vs. Time

Figure 16 shows a graph of the predicted solar array output in watts as a function of time. The most critical portion of this scenario is when the Juno spacecraft makes its planned turn for the JOI burn. This turn requires that the spacecraft solar panels be turned nearly 90 degrees from the sun line. This results in no solar input on the solar arrays and, hence, that Juno spacecraft run on battery power only.

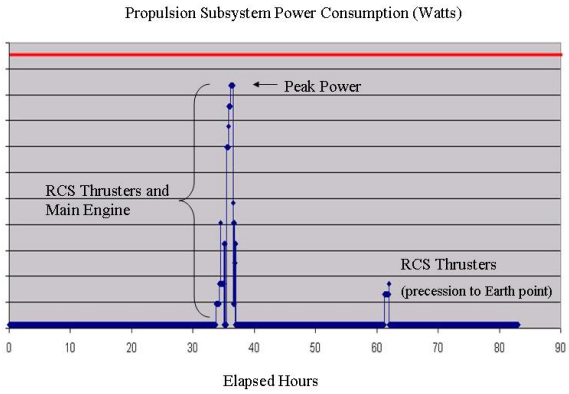


Figure 17 – Propulsion Subsystem Power Consumption vs. Time

Figure 17 depicts a graph of the amount of electrical energy in watts that the propulsion subsystem consumes as a function of time during the JOI scenario. This subsystem is the most power hungry subsystem during the scenario so Juno project engineers are concerned about the batteries being able to support the critical maneuver. The horizontal red line shows the system requirement for maximum power consumption that the propulsion subsystem should stay under for the duration of the scenario.

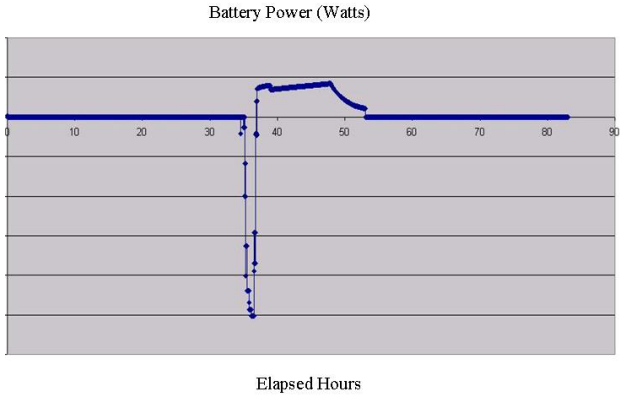


Figure 18 – Battery Power vs. Time

In Figure 18, the battery power supplied as a function time is depicted. When the Juno spacecraft turns for the JOI burn it is running entirely on battery power. This graph helps engineers understand the amount of stored energy that the batteries must provide in order for the spacecraft to survive the maneuver. It should be noted that predictions that lie

below the x-axis indicate that the battery is discharging and predictions above indicate that it is charging.

9.0 CONCLUSIONS

In FY 2006, JPL's Research and Technology Development (R&TD) program undertook an effort to develop and pilot formulation phase model-based engineering design capabilities by "shadowing" two deep space missions. Since much of the activity in this phase center on system engineering, the initiative focused on developing its Model-Based System Engineering (MBSE) capabilities.

The pilots demonstrated how the traditional document-based functional analysis can be enhanced through the use of a computer-aided model-based systems engineering tool. Model-based tools and techniques such as these improve system requirements and design by providing automated audits of the model's consistency, completeness and integrity.

In the model-based approach, system engineers focus on creating descriptive models, rather than writing documents. The models now become the repository for system engineering knowledge so that artifacts, such as gate product documents, can be automatically produced rather than written. When integrated with a web-based inter-application communication executive, one can select and transform system engineering information to enable other kinds of analyses such as simulation-based risk and performance assessments.

Shortcomings in functional analysis' ability to represent dynamic system behavior were overcome with the introduction of mission planning, performance simulation and state analysis. Synthesizing functional and state analysis highlighted the complementary nature of these two model-based systems engineering methodologies and promises to yield significant benefits, including:

- Better understanding and documentation of designed behavior;
- Earlier identification of unexpected design challenges;
- Improved traceability to developed software; and
- More robust fault protection in the designed system.

Integrating the state analysis schema into a general-purpose model-based systems engineering tool provides for traceability between functional models and behavioral models, and allows for easy navigation between these different but related domains.

The task also demonstrated the utility and applicability of time-based performance and resource simulations and mission planning in providing early verification of system requirements.

ACKNOWLEDGEMENTS

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

References herein to any specific commercial product, process or service by trade name, trademark, manufacturer, or otherwise does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

REFERENCES

[1] Robert Shishko, Robert G. Chamberlain, *NASA Systems Engineering Handbook*, NASA Publication SP-6105, Jet Propulsion Laboratory, 4800 Oak Grove Drive, Pasadena, CA 91109, second printing April 1996.

[2] Department of Defense Systems Management College, *Systems Engineering Fundamentals*. Defense Acquisition University Press, Fort Belvoir, VA., 2001.

[3] Vitech Corportaion Web site: http://www.vitechcorp.com.

[4] 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.

[5] Dvorak, D., Rasmussen, R., Reeves, G., Sacks, A., "Software Architecture Themes in JPL's Mission Data System," *IEEE Aerospace Conference Proceedings*, March 2000.

[6] Bennett, M., Rasmussen, R., and Ingham, M., "State Analysis Requirements Database for Engineering Complex Embedded Systems", *15th Annual International INCOSE Symposium*, Rochester, NY, July 2005.

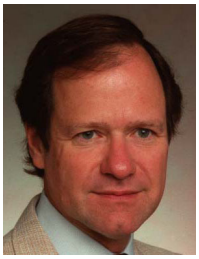
[7] D. Y. Stodden, G. D. Galasso, "Space System Visualization and Analysis Using the Satellite Orbit Analysis Program (SOAP)," IEEE Aerospace Applications Conference Proceedings Vol. 2, Aspen Colorado, 1995.

[8] Kordon, M., and E. Wood, "Multi-Mission Space Vehicle Subsystem Analysis Tools," IEEE Aerospace Conference Proceedings, Big Sky, MT., March 2003.

[9] The Eclipse Foundation Web site: http://www.eclipse.org.

BIOGRAPHY

Mark Kordon *is the Technical Group Supervisor for the Modeling and Simulation Technologies Group, and Task Manager for Integrated Spacecraft Analysis Tools at the Jet Propulsion Laboratory. His research interests include modeling and simulation techniques, evolutionary computing, multi-agent systems and space systems. He received his degree in Computer and Systems Engineering from Rensselaer Polytechnic Institute. Mark managed the work described in this paper.*



Steve Wall *is a Principal Engineer at the Jet Propulsion Laboratory, California Institute of Technology, where he manages programs for the Strategic System Technologies Program Office. Steve has participated in seven major space missions in design teams, science teams, operations teams, and management. Current research interests include advanced system design, concurrent engineering, and other rapid design methods. He holds a Masters in Optical Engineering from the University of Rochester and a BS in Physics from North Carolina State University. For his past work Steve has published over 85 papers in the scientific and technical literature and in the popular press and has been awarded the NASA Exceptional Achievement Medal, the Exceptional Service Medal, and six NASA Group Achievement Awards.*

Dr. Henry Stone *Dr. Stone is the Section Manager of the Systems Engineering Section at the Jet Propulsion Laboratory. He began his career at JPL in 1987 and worked on a number of robotics technology programs. In 1992, he became the Technical Manager of the Mars Pathfinder Rover's Control and Navigation Subsystem. Between 1997 and 1999 he was a sub-lead on several Mars Mission studies including the 2003 Athena Rover and the 2004 Large Lander Study. In June of 2000 he became the Project Element Manager for the Mars Exploration Rover Avionics Subsystem. During operations he led MER's Spacecraft/Rover Engineering Team. Dr. Stone obtained his BS, MS, and PhD in Electrical and Computer Engineering from Carnegie-Mellon University in 1981, 1983, and 1986, respectively.*

William Blume *is the Technical Group Supervisor for the Mission Engineering and Planning Group at the Jet Propulsion Laboratory. The group is responsible for developing plans, strategies, and end-to-end mission timelines for JPL space missions. He was recently the mission design manager for the Deep Impact project that impacted comet 9P/Tempel 1 in 2005. Bill has BS and MS degrees from Purdue University in aeronautics and astronautics. He planned the approach for integrating mission activity plans with the subsystem modeling tools.*

Joseph Skipper *is a member of the technical staff in the Exploration Systems Concepts Group. He is currently engaged on the Constellation program on both the Command, Control, Communication, and Information (C3I) team, and the Lunar Operations Simulation Modeling team. He holds a Ph.D. degree from Texas A & M University.*

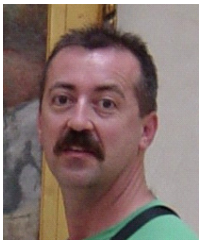
Dr. Michel Ingham *is a member of the technical staff in the Flight Software Systems Engineering and Architecture Group at the Jet Propulsion Laboratory. His research interests include model-based methods for systems and software engineering, software architectures, and spacecraft autonomy. He earned 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. He was responsible for architecting the integration of Functional and State Analysis.*

Joseph Neelon *is a member of the technical staff in the Mission Engineering and Planning Group at the Jet Propulsion Laboratory. He received his Bachelor of Science in Aerospace Engineering from Penn State University, University Park and his Master of Science in Astronautics from the Massachusetts Institute of Technology. Joseph analyzed the geometries and assembled the activity plans described in this paper.*

Ron Baalke *is a member of the technical staff in the Mission Engineering and Planning Group at the Jet Propulsion Laboratory. Ron received his Bachelor of Science in Computer Science from California State University, Sacramento and his Master of Science Degree in Computer Science from California State Polytechnic University, Pomona. Ron researched and entered the spacecraft requirements into the system engineering tools and generated the gate products described in this paper.*

James Chase *is a member of the technical staff in the Surface Systems System Engineering Group at the Jet Propulsion Laboratory. He received his Master's Degree in Aeronautics and Astronautics from in MIT. Jim helped with the Juno requirements and state analysis efforts.*

David Hanks *is a member of the technical staff in the Modeling and Simulation Technologies Group at the Jet Propulsion Laboratory. He received his Bachelor of Arts in Mathematics and Bachelor of Science in Physics from the California State University, Fullerton. David is the Integrated Spacecraft Analysis Tools Development Lead. He was responsible for the simulation results described in this paper.*



Jose Salcedo *is a member of the technical staff in the Modeling and Simulation Technologies Group at the Jet Propulsion Laboratory. Jose received a Bachelor's of Arts in Physics from Occidental College('82) and a Master's Degree in Computer Engineering from the University of Southern California ('84). Jose implemented the binary comparison expression code described in this paper.*

Benjamin Solish *is a student at the University of Washington in Aeronautics and Astronautics working on his MS degree. His research interests include space systems engineering, spin orbit resonance, and control theory. He earned his B.S. from MIT in Aeronautics and Astronautics. He was responsible for implementing the integration of Functional and State Analysis.*

Mona Postma *is a member of the technical staff in the Modeling and Simulation Technologies Group at the Jet Propulsion Laboratory. She received her Bachelor of Science in Computer Science from California State Polytechnic University, Pomona. Mona researched and entered the spacecraft requirements and designs into the system engineering tools and performed the requirements verification.*

Richard Machuzak *has been a member of the technical staff in the Mission Engineering and Planning Group for the past 5 years, and is the Mission Designer on the Space Interferometry Mission. Rich was responsible for researching and entering much of the spacecraft data in the systems engineering tools.*