Identification Page

Paper Tit le: Implementation of System Requirements Models for Space Missions

Authors: S. D. Wall
J. C. Baker
J. A Krajewski
David B. Smith

Business Affiliationn (all authors):
Jet Propulsion Laboratory,
California Institute of Technology

correspondence address:
Stephen 1). Wall
Mail Stop 301-230
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena CA911 09 USA

Phone: (818)354-7424
Fax: (81 8) 393-0028
Email: Stephen.D.Wall@jpl. nasa.gov

Stephen 1). Wal I is Subdomain Owner for Mission and System Design at Jet Propulsion Laboratory. He has previously led the Advanced Projects Design Team ("Team X") and managed mission operations for a number of JPL projects. Steve is also Deputy Team Leader for the Cassini (Saturn) Radar Science Team and has participated in several ground and flight system experiment design teams for JPL and NASA. He previously was a camera design engi ricer for NASA's Langley Research Center in Hampton, VA. He has a BS degree in physics from Norh Carolina State University and a MSEE in Optical Engineering from University of Rochester.

John Baker is Process Owner for Systems Engineering at JPL. He has been instrumental in the development of process-based system engineering at JPL. Previously, John designed digital systems for the Shuttle Radar Laboratory and participated in instrument designs. He holds a degree in physics from Colorado State University.

Joel Krajewski obtained a BS in Engineering Physics at U.C. Berkeley and joined the MITRE Corporation. At MITRE, he worked in various roles on missile defense systems, culminating in the position of Associate Project Leader for Air Force Theater Missile Defense Command, Control, Communications, and Intelligence. In 1995, he returned to U.C. Berkeley to obtain an MS in Mechanical Engineering. Mr. Krajewski joined the Jet Propulsion Laboratory in June 1997 as a Senior Engineer in the Flight Systems Section,

David B. Smith was graduated from West Virginia University in 1962 in Aerospace Engineering. He received a Masters degree from University of Southern California in 1965 in Aerospace Engineering. 1 Ie has been with JPL's planetary program since 1971, beginning with Mariner Venus Mercury, Viking, then Voyagers 1 and 2 and Galileo. After two Shuttle Radar Laboratory missions he is currently the team leader for JPL's product reengineering efforts.

Implementation of System Requirements Models for Space Missions

S. D. Wall
J. C. Baker
J. A Krajewski
David B. Smith

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena CA 91109 USA

Abstract -As a part of it restructuring of the space mission design process, the Jet Propulsion Laboratory is investigating a model-driven concept for capturing system-level requirements for space misions. Anticipated advantages include (1) earlier achievement of system design maturity; (2) earlier detection of system-level problems through virtual test; and (3) enabling of mission/system trades throughout the design phase. Model-driven systems enable rapid evaluation of changes and, it is hoped, 'significantly shorter development periods and less rework.

1. INTRODUCTION

Prior to 1990, space missions were proposed, designed and operated under a pardigm of performance maximization. The frontier of space exploration seemed infinite, and the demand for technology development encouraged that paradigm. Toward that end, space missions were conceived that produced maximum exploration and maxi mum technology development; economics and productivity were secondary.

The 1990s brought an environment that was significantly different. The end of the cold war and the coincident focus on federal budget balancing brought with them an era of practicality applicable to, among other fields, space exploration. Science and technology insertion were still required; but in addition productivity and maximization of the science-to-cost ratio were of concern. NASA introduced the "faster, beter, cheaper" mantra/ both full cost accounting and total-life-cycle cost accounting were required. In short, scientists and engineers. are now asked to design and build systems where cost and schedule may be fixed, or near-fixed, parameters in the design space. A key effect of this new design environment is that the technical requirements at the system and subsystem level must remain more flexible throughout the design cycle.

After some deliberation, JPL's reengineering of product development centered itself around six basic themes: (1) . The overall approach to satisfaction of these themes is addreseed in Smith (1997). The focus of the present paper is on the MSD contribution to the DNP stretch goals, which is concentrated on the achievement of early test of the system requirements themselves We will describe a methodology that captures system requirements in software models rather than in documents.

2. EXECUTABLE REQUIREMENTS

At the highest level, requirements for a traditional project are stated in terms of science (or other sponsor) objectives: to visually map a planet's surface, to determine surface modification processes, etc. Such objectives are typically refined into requirements on a spacecraft (e. g., the mass it must carry, data it must store) requirements on the mission (e. g., orbit characteristics, data to be acquired), and any other major elements. These in turn are partitioned into systems within these elements, subsystems

within the systems, and so on until the requirements are stated at a low enough level to be handled by a single engineering team. Traditionally, such refinement is captured and communicated on paper, in documents which are used to define commitments by the designers. Committments are [hen negotiated against resources, usually using the same documents, until an agreement is made. For large space missions the flowdown of, and commitment to, ful I set of requirements to a buildable level can occupy many workmonths and several calendar years to complete.

Doeument-driven design, as this methodology is termed (Baker et al, 19xx), suffers from the limitation that its control mechanism is slow. Captured in such a flat form, comprehending tile inter-dependencies of subsystem requirements well enough to formulate an effective change, let alone braving the document adjustment process, can be daunting. Typical I y, the set of requirements documents are understood in full by a handful of people, at best. The addition of requirements tracing software that manages requirements documents in electronic form and identifies requirements that are "children" of higher level requirements has relieved the paperwork burden but has largely left the comprehension and change control aspects untouched.

A second difficulty with document-driven design is that it does not lend itself to testability. High-1evel science objectives are the ultimate customer requirement, yet it is difficult to determine whether a change still meets them or not. For example, to determine if a telecommunications subsystem's 10-watt transmitter is compatible with a scientist's desire to make a map of Venus at 100-m resolution in less [ban one year needs considerable calculations. To evaluate the impact on that desire of a 10% increase in transmitter mass is harder still.

For both these reasons, J PL has undertaken to partially or wholly replace the document-driven requirements process with something better. Following an idea from Purvis et al., 19xx and expanded by Smith, 1997, we have begun implementation of a requirements system based on a set of cross-cutting requirements models ("requirements models") that state spacecraft-level requirements (referred to as Level 3 requirements) in an executable form. The desired advantages of this new methodology are (1) it should permit an earlier and easier achievement of a level-three design commitment between system engineer and design engineers; (2) it should enable testing of requirements against some quantitatively stated objective and thus permit earlier detection of system-level problems through virtual test; and (3) as the implementation of the design proceeds, it should continue to allow mission/system trades as a way of evaluating potential changes to either system design, mission objectives or subsystem design.

3 . IMPLEMENTATION .

Relationship to Current Tools

Models don't replace engineers -- rather, when used shrewdly they help the design team to do its job faster and better by catching some system-level problems earlier in the design cycle and affording quicker and/or more careful requirement allocations to be explored. They can be viewed as combining in one place the features of several tools we already use for mission and system design in order for the team in general, and the system engineer in particular, to capture, understand, and improve system-level behavior.

Often, activities during a particular spacecraft mode are such that the demand for system-level resources are fairly constant, or at least easily boundable. On the other hand, there can be some spacecraft modes that represent significant variations of demand in time that are hard to assess in a simple mode-based analysis. An example from a current JPL avionics design project, X2000, is a trajectory correction maneuver (aka, 'turn and burn'). This maneuver consists of a sequence of modes: cruise - turn - main engine burn - turn - cruise. An excel-based analysis of this manuever was performed by the X2000 system engineer in May 1997 with

two results: (1) the power requirement is challenged by this manuever, but (2) it was hard to see the relative timing of various events, The partial satisfaction provided by this analysis approach constitutes one of the motivates for a more detailed tool.

XCUT models simply combine many of the features of spreadsheets, block diagrams, and mission scenario timelines into a fourth tool: a dynamic system model (see figure 1). XCUT models depict the basic subsystem partitioning and interactions (from functional block diagrams), embue these diagrams with the subsystem modes, states, and interactions relevant to utilization of system-level resources (from budget a llocat ions), and drive the diagrams through a script derived from mission scenarios to show gross-level system behavior over time.

requirements balance is impacted by a subsystem having a different capability than its initial requirement allocation. Though perhaps confusing at first, this flexibility is no different from the flexibility we use today in Excel spreadsheet budgets.

X2000 XCUT Models

The initial version of the XCUT model for X2(X)O is taking shape. We are currently using the tool "Foresight" to implement the XCUT models. Figure 2 depicts the goal model structure.

The X2000 system engineer identified power, databus, mass memory, and CPU as key system-level resources to be managed during the design process. Of these, power and databus requirements are the first system-level resources-we chose for the XCUT

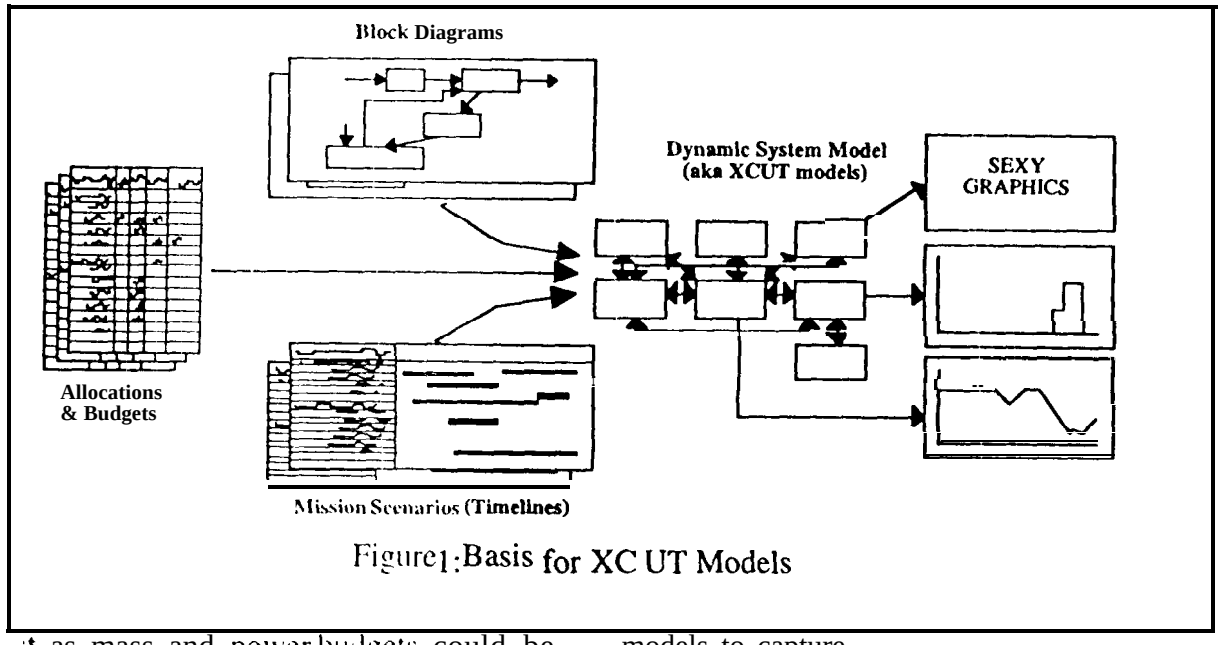


Figure1:Basis for XC UT Models

J st as mass and power budgets could be interpreted as either top-down requirements (early in a project), bottom-up capabilities (later in a project) or some deftly-handled combination of both, the interpretation of the outputs of the XCUT models depend on how the input parameters are interpreted -- validation of internal consistency amongst Level 3/4 requirements, validation of consistency amongst subsystem capabilities, or, mixing requirements and capabilities, an assessment of how the overall system

models to capture.

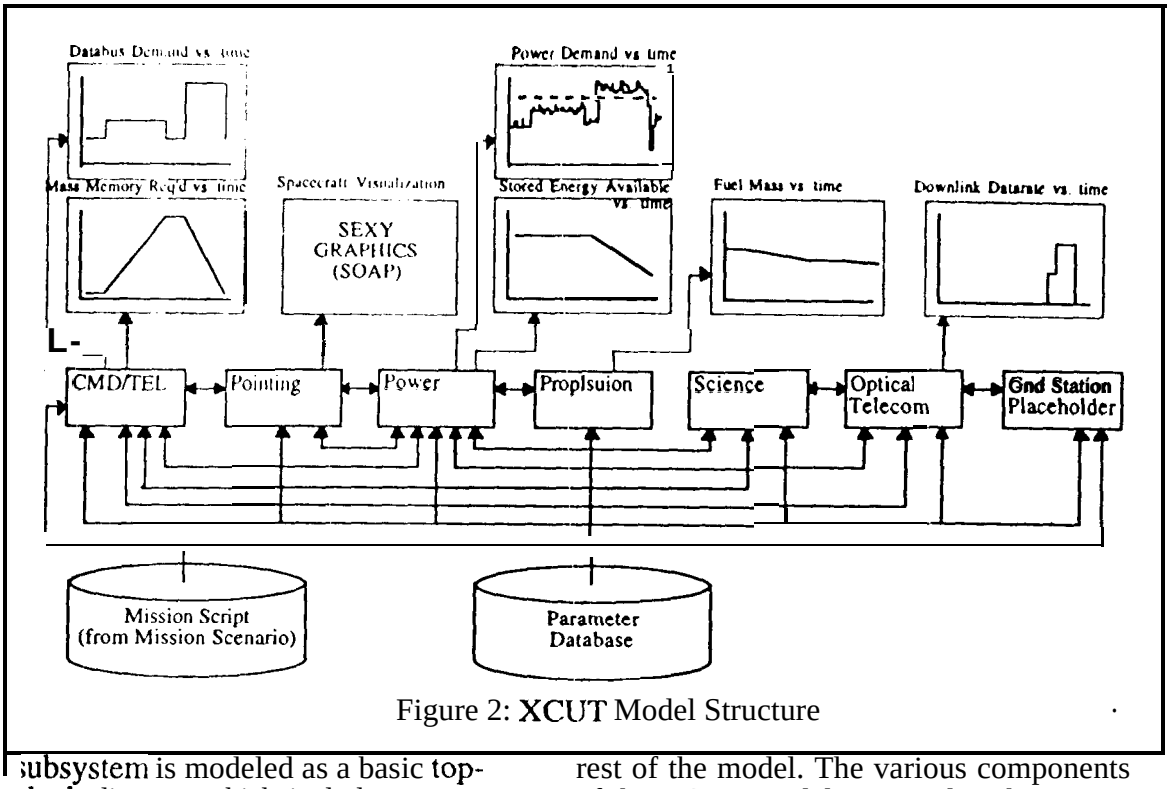


Figure 2: XCUT Model Structure

Each subsystem is modeled as a basic top-level block diagram which includes state transition diagrams to capture subsystem states (e.g., 'off','warming up', 'on', 'standby', 'communicating', etc.). Each subsystem is characterized by a set of top-level parameters, often associated with particular states, of the sort [hat are specified in resource allocation sheets (e.g., datarate, power draw, warm-up delay, etc.).

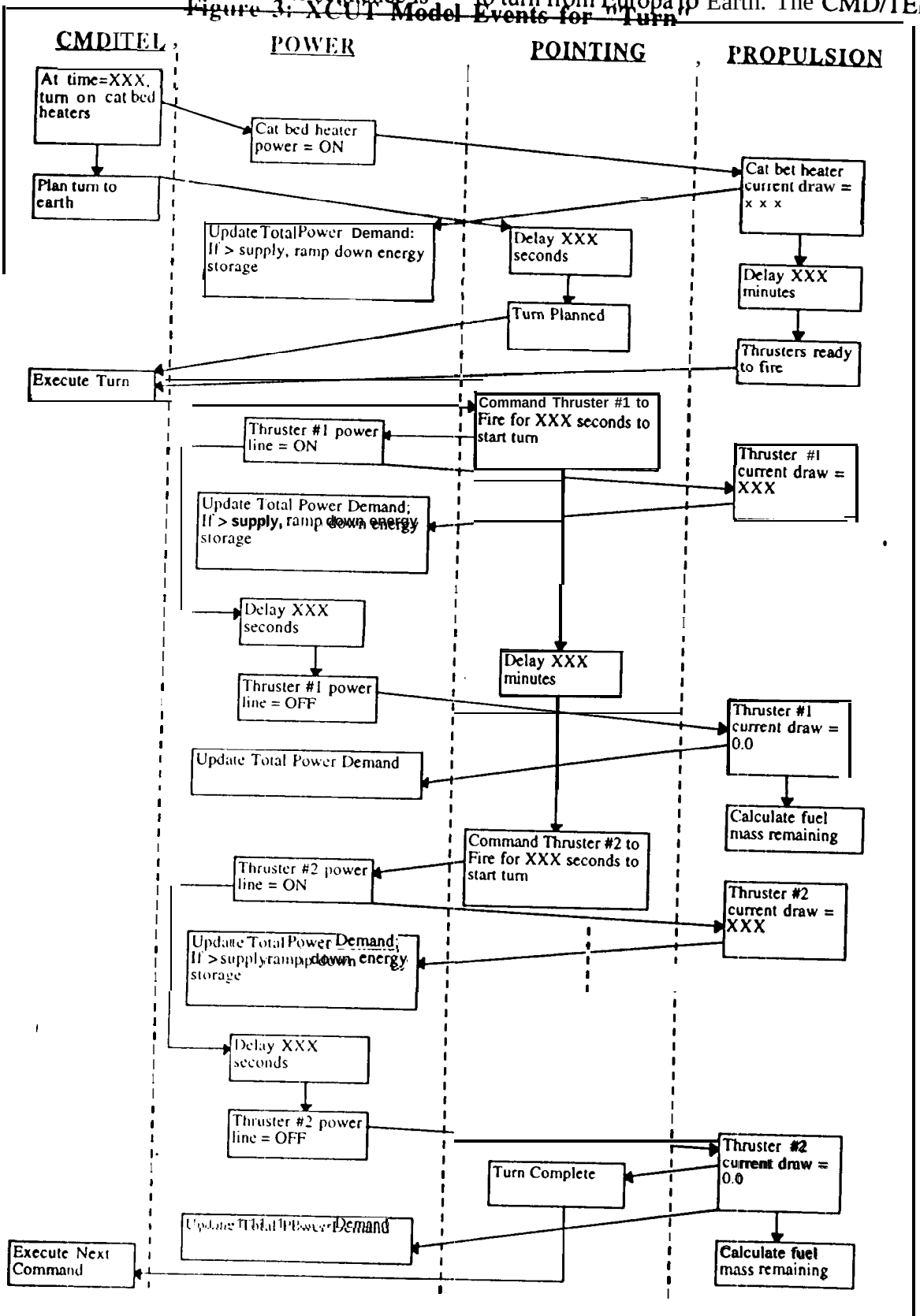
The model reads in initializing data and a script derived from a mission scenario. Initializing data include subsystem parameters (e.g., power draw and databus demand for each state), initial states of each subsystem at the start of the scenario, and so forth. The script consists of two sets of inputs: (1) commands loaded into a sequence in the cmd/tel model to be executed during the first portion of the scenario, and (2) commands loaded into the ground station model that are to be uplinked to Command/Telemetry (via telecom) during. the scenario for later execution,

After the models are initialized, the user starts the simulation. The command/t elemetry model issues commands stored in its sequence at the appropriate times to drive the rest of the model. The various components of the XCUT models respond to the commands and interact. "

A concrete example maybe helpful here: figure 3 below shows the interactions among the XCUT models to perform a turn, followed by explanatory text.

Assume the first command in the sequence is to turn from Europa to Earth. The CMD/TEL

Figure 3: XCUT Model Events for "Turn"



model issues a command to Power to turn on catalyst bed heaters. The Power model, upon receipt of this command, sets the value of a power line running into the Propulsion model to be ON. In response, the Propulsion model sends back to the Power model a parameter reflecting the current that the cat bed heaters are now drawing. The Power model then updates its summation of the overall power demand accordingly. If the power demand exceeds the supply, then the energy storage reserve is drained over time to support the load.

While the beds are heating, CMD/TEL commands Pointing to plan a turn (o Earth. After a delay, the Pointing model sends a message to CMD/TEL that the tum has been planned. The CMD/TEL model commands Pointing to execute a turn after a delay accounting for the warm-up time required by the cat bed heaters and afte r receiving the 'turn planned' message from Pointing (whichever comes lint). The Pointing model then sends command to the valve drive electronics (located in the Power model) to open a thruster for a length of time. In response, the VDE changes the value on a power line to the thruster in the propulsion model to open the thruster, The Propulsion model sends back to the power model the current draw required to open the valve.

After a delay accounting for the coast period of the turn has elapsed, the Pointing model commands a second thruster to fire to stop the turn. A time delay is then encountered to account for settling time into the new attitude. After this delay, the Pointing model sends a message to the CMD/TEL model stating that the turn has been executed. The CMD/TEL model then executes the next command in the sequence.

In terms of model outputs, the Propulsion model displays a time-history of fuel consumed, which is updated each time the fuel flow rate changes. The Power model sums up the overall power demand at any given point in time, compares it [o the DC power supply (less conversion and distribution losses), and continuously updates its di.sill.i~\ of power demand, power margin, and % energy reserve available.

When demand exceeds supply, the difference is covered by draining an energy reserve (less a discharging loss), and the energy supply display ramps down accordingly.

Downlink, uplink, science collection, and other spacecraft modes are modeled in a similar fashion. In this way, the XCUT model executes a script derived from a mission scenario, and accounts for the demand for system-level resources either in a requirements sense (if parameter values representing requirements are used), or in a capabilities sense (if parameter values represent capabilities are used).

Further, when a subsystem engineer proposes a capability that varies from the initial top-down requirement, the models can facilitate the assessment of the resulting impact to the system-level requirements balance.

Interaction between Cross-Cutting System-level Models and Subsystem Design Models

For years, subsystem engineers have built detailed, physics-based computer models of subsystems in order to better estimate performance and surface technical design or operational issues. Examples include attitude control system models built in MATLAB, mechanical models built in ProE, and so forth. Generally, a CogE builds and uses these models to flesh out hardware/software design and, in some cases, to produce design files that can be used for fabrication directly.

Although the concept of integrating all these detailed, physics-based models together into a complete spacecraft model (aka, a 'virtual spacecraft') may at first seem appealing, the practical impediments are legion:

- profoundly complex software interfaces
- computer power required
- difficulty& turnaround time to make changes that affect multiple models

Hence, the main goal of partitioning the modeling activities into XCUT and subsystem design (SSD) models interfaced through a Parameter Database is to shrewdly

select key design parameters for each subsystem, and then exercise a system-wide model driven only by this relatively small set of characterizing parameters.

There are (at least) 3 distinct ways in which parameters shared by XCUT and SSD models could be used. with notional examples:

(1) 'Top-down design'

Assume a situation in which system-level requirements have been established and balanced by the system engineer using the XCUT models. The resulting requirements parameters are then uploaded to the PDB for the team's use.

Assume a subsystem engineer (say, the Telecom engineer) is using an SSD model to develop a subsystem design, and there is a top-down requirement levied on the subsystem (e.g., a transmit datarate). The engineer's model would capture the physics of the problem: power consumption and losses, beam spread, distance to receiver, pointing accuracy. performance variation with temperature, etc. Ultimately, one of the outputs of this model is a predicted datarate achievable under different conditions.

The engineer's SSD model would, when run, download the required datarate from the PDB and plot it in [he same chart as the predicted achievable datarate, thereby allowing instant, obvious comparison of predicted performance versus a top-down requirement. The SSD model would do no calculations using the required datarate, merely use it in a plot for comparison purposes.

(2) 'Boundary conditions for subsystem design'

Consider a Power engineer who's task is to design a power system, and he/she uses an SSD model that captures the physics of power generation and storage: solar array efficiencies, sun-spacecraft-planet geometry, transmission losses, conversion losses, thermal behavior, battery discharge / recharge mechanisms and efficiencies, transient behavior, etc.

The system engineer would have run the XCUT models with mission scripts and obtained power-demand profiles from the simulation. These profiles would have been based on the allocated power demand for each state within each subsystem, as played out by the mission scenarios. The power engineer can then use the power demand profiles from the XCUT models to design against.

(3) 'Assess Impact of Subsystem design variation from original requirement'

Assume that the original requirements balance achieved by the system engineer using the XCUT models is being challenged; in particular, that the Telecom engineer, using his/her SSD model, is arriving at design solutions that meet the datarate requirement but break his/her power requirement. The Telecom engineer loads a "Current Best Estimate" capability value for power into the PDB. The system engineer, perhaps with the design team in attendance, then runs the XCUT models again, with the CBE power capability value instead of the requirement, to assess the impact on the overall system requirements balance. If the perturbation is minor, perhaps the design team will decide to relax the power requirement on Telecom. If the impact is significant, then the team can explore a number of possible options: increase energy storage, temporarily turn off other power-consuming components, and so forth. Use of the graphical XCUT model both speeds up the initial assessment of system-level impact and facilities communications among the team as to the causes and potential solutions to system design problems.

The importance of the SSD models in these example is that they are a primary tool for the subsystem engineer to provide the 'technical pedigree' of the characterizing parameters used in system-level trades.

The importance of the XCUT models in this example is that the interrelationships among all the subsystems' characterizing parameters are captured and exercised with quick turnaround (measured in minutes) against a

mission scenario. The XCUT models are a primary tool for the system engineer and the design team to work through system-level trades.

At this point it should be acknowledged that SSD models will probably not be the only source of 'pedigreed' characterizing parameters. For instance, if the design calls for flying an existing piece of hardware, then parameters derived from test data, not an SSD model. would probably be used. It is likely that any given project will have a need for detailed SSD models of some, but not necessarily all, subsystems.

Similarly, any given project will probably utilize some aspects of the XCUT models more than others, depending on the design team's assessment of key technical issues to be resolved.

4. SUMMARY AND CONCLUSION

A first implementation of a model-driven system design concept has been completed, With this first set we intend to pilot JPL projects through development of requirement> at level 3 for spacecraft and flight instruemntation, carefully measuring progress against expectations and looking for lessons to learn. Expected technical benefits include better expression of complex information. and clearer and less ambiguous definitions of design. Expected programmatic benefits are earlier achievement of system design maturity, earlier detection of system-level problems through virtual test; and enabling of mission/system trades throughout the design phase.

5. ACKNOWLEDGEMENTS

The Develop New Products Project is indebted to its former leader, Mike Sander, for guidance and leadership, and to its design team for their dedication and creativity. This The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

6. REFERENCES

Michael Hammer and James Champy, Reengineering the Corporation, New York: Harper Business, 1993.

M. Golumbeck, "The Mars Pathfinder Mission", JGR *Planets* 102, pp. 3953-3965, February, 1997.

L. Baker, P. Clemente, B. Cohen, L. Permenter, B. Purves and P. Salmon, "Foundational Concepts for Model Driven System Design", INCOSE Model Driven Systems Design Working Group, 1997.

Smith, D. B., "Reengineering Space Projects", Computer Tools, Systems Engineering and Competiteveness Symposium, Paris France, March 5- 7, 1997