

A Rapid, Flexible Approach to Tradespace Definition and Exploration

André R. Girerd
 Jet Propulsion Laboratory
 California Institute of Technology
 4800 Oak Grove Drive
 Pasadena, CA 91109
 818-354-1771
 andre.r.girerd@jpl.nasa.gov

Abstract^{1,2}—This paper provides an overview of the Mission Tradespace Tool (MTT), a methodology and software framework developed to improve JPL’s early design process by offering a rapid, structured, and inexpensive way to identify feasible space mission design architectures from a wide array of candidate architectures.

There has been a growing consensus at JPL that to improve the quality of service offered to design customers it is desirable to explore a wide tradespace of candidate architectures prior to forming a conceptual design baseline. This paper describes the rationale behind the MTT’s approach to meet this need. Notable features of the framework are introduced and explained.

Mission Tradespace Tool (MTT) addresses this problem by offering an approach to rapidly and inexpensively create and analyze a mission’s tradespace.

The MTT is geared for rapidity (<1 week) and flexibility (multiple mission types), with appropriate level of detail for early project planning. The first objective is to capture the design space in a structured, consistent, and archiveable manner. This involves the input of various trade options and associated technical parameters. The second objective is to identify the feasible regions of the design space along multiple axes of cost, risk, and performance. This way, infeasible regions of the tradespace can be avoided and promising architectures identified with proper context. Thousands of architectures (option combinations) can be evaluated and analyzed. Figure 1 shows the basic flow of the MTT process.

TABLE OF CONTENTS

- 1. INTRODUCTION 1
- 2. DATA SOURCES 4
- 3. TERMINOLOGY 4
- 4. USER INTERFACE 5
- 5. VIEWS AND FEATURES 5
- 6. CONCLUSIONS 7
- REFERENCES 8
- BIOGRAPHY 8
- ACKNOWLEDGMENTS 8

1. INTRODUCTION

A growing body of recently published research addresses techniques for quantitatively exploring large design tradespaces populated by architectural candidates produced by various model-based methodologies [1],[2]. Unfortunately, the time and cost necessary to assemble a suite of integrated, customized models can often exceed the resources available to space mission projects in the early conceptual design phase. Thus in practice, despite the theoretical and demonstrated usefulness of such techniques, project leaders face immense pressure to downselect to a single baseline without rigorously establishing the desirability of their choice within a suitable context. The

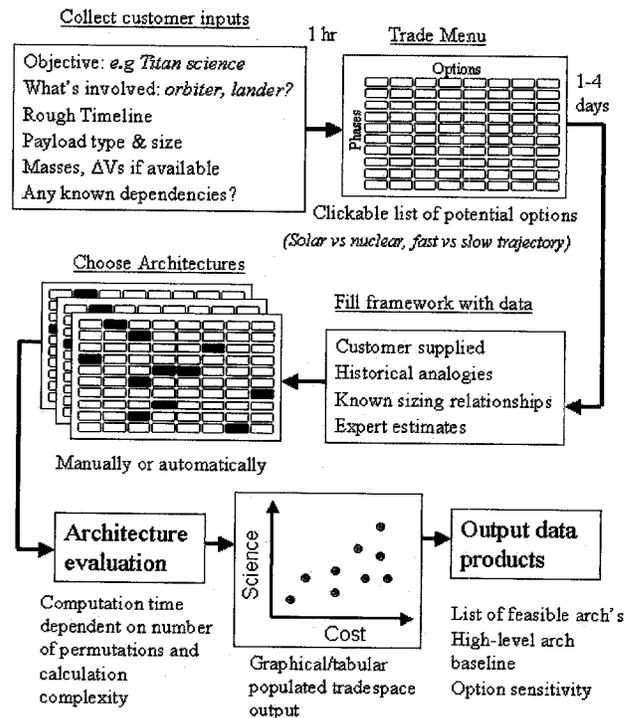


Figure 1: MTT Process Flow

¹ 0-7803-8870-4/05/\$20.00© 2005 IEEE

² IEEEAC paper #1503

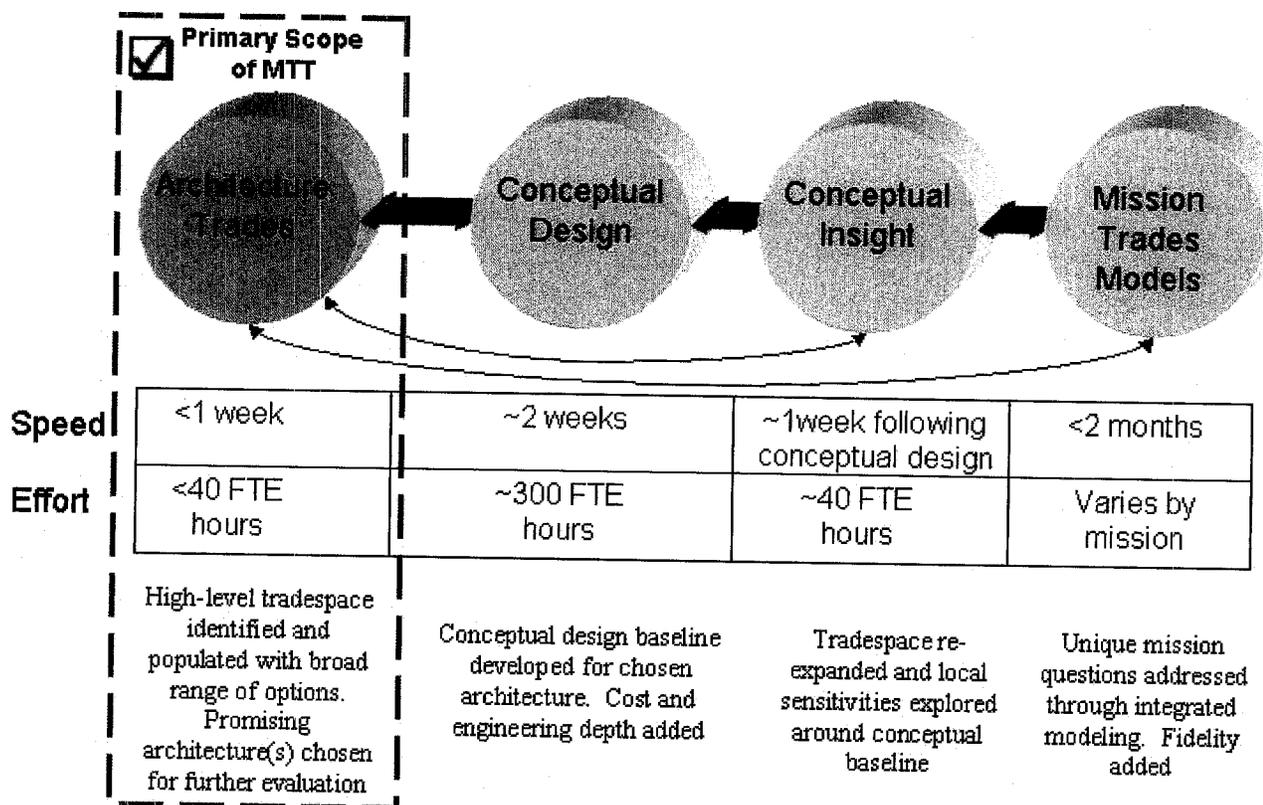


Figure 2: MTT Scope within Greater Early Design Process

The MTT is envisioned as the first part of a larger, iterative, design process wherein high level tradespaces are created and winnowed, one or more promising architectures are selected and analyzed in more detail, and detailed design excursions are made within a limited region for localized fine-tuning. Figure 2 illustrates the role of MTT within this system design process vision. Although the MTT inhabits the earliest part of the greater process, it takes advantage of feedback opportunities for successive cycles. With each cycle, fidelity improves, and mutual benefit is gained between architecture trades, conceptual design and subsequent stages. Because the MTT offers architectural trade results so quickly and inexpensively, it can provide useful broad context for later design stages well within their cycle time.

Scope

The MTT is not purposed to capture elusive, tightly nuanced, poorly understood design interactions for missions with extreme sensitivity to assumptions. The scope is geared for an intentionally rough, first cut pass through a wide tradespace to discard clearly infeasible designs and identify a region with some promise for further investigation. Within this context, any notion of optimization is inappropriate. An architecture that performs better than its competitors for any given criteria could easily get displaced or reduced in desirability by some new or

unaccounted system or mission perturbation, be the reason technical or programmatic. Inhabiting the volatile world of early mission design where assumptions and entire mission concepts change rapidly make this difficulty par for the course. The emphasis is firmly on rapidly assembling only the currently perceived first-order driving relationships. This necessitates a high-level, lower fidelity approach. Yet higher fidelity does not always imply higher accuracy, but almost surely implies greater effort. Indeed, the goals of flexibility and rapidity are mutually exclusive of high fidelity. The trick is to incorporate just enough fidelity to ascertain gross feasibility and have some confidence in the relative impact of major design choices.

Drawbacks of Existing High-level Tradespace Methods

Current quantitative methods for performing high level systems-oriented tradespace analysis have not operationally matured for the fast-paced tempo of the earliest design phase, before a conceptual design baseline has been selected or identified. For broad, shallow tradespace investigations, universities have proven particularly adept [3], but only when the types of desired inputs have not changed over the timeframe required to construct the supporting software.

Theoretically, large teams of dedicated, well-coordinated tradespace model builders and integrators could handle most needs within a severely limited timeframe, but such

resources are rarely allocated so early in the process. The time to perform an analysis depends on the availability and skills of the supporting engineers, as well as the particular nature of the problem. This time is minimized if fidelity creep can be kept under control. Regardless, the process is never done in real time, and infrequently (at best) within the desired decision cycle.

Typically, design relationships are encoded into software with spreadsheets being a particularly popular option. In the absence of any organizational standard or encouraged procedure, a wide variety of different implementations can arise from cognizant engineers with varying levels of programming and spreadsheet skills. Inevitably, some trades are resolved and new trades emerge. Sometimes the existing code or spreadsheet can be modified to accommodate the revised tradespace, but since the modifications were not anticipated during the initial software construction, it frequently becomes messy. This is exacerbated by the time constraints of the early process, which do not favor leisurely, systematic reappraisals of existing coded logic. Any change in personnel can also complicate the process. Too often, the only viable solution is to start from scratch all over again.

The Importance of a Flexible and Structured Framework

Offering quantitative tradespace exploration capability through the operational difficulties of early architectural design requires the ability to add, subtract, and evaluate new system level trades on a frequent basis without having to revamp the basic underlying structure of the trade tool. This places a premium on a framework that emphasizes flexibility. Additionally, since a framework can be standardized, anyone familiar with its use should readily understand previous tradespace instantiations. This encourages reuse both within an evolving architecture study and between different studies that share some commonalities. Furthermore, a structured framework with significant flexibility to accommodate a range of missions should produce, over time, a valuable record of common trades, and act as a standard repository of system engineering information and assumptions. Such a collection would be invaluable for data mining and continuous process improvement.

A flexible software structure geared explicitly for space mission trade exploration offers a major advantage in that it reduces the organizational design burden otherwise required when crafting a single use spreadsheet. Many of the commonly needed features and necessary bookkeeping functions are already present, built-in. This relieves the user of the preponderance of software overhead and allows a more uninterrupted focus on system engineering, increasing efficiency and speeding the process. Furthermore, the framework with an evolving feature set can be tried, tested, and honed over time.

The exercise of populating a high-level tradespace to identify feasible and infeasible design architectures is less valuable if it cannot produce results before embarking upon the more detailed conceptual design process centered around a single architecture. A slow tradespace exploration process will tend to lag behind the formation of a technical baseline, and although it provides useful context (ex post facto) for the design point it will not be timely enough to assist in the point's selection. A truly viable rapid early tradespace exploration method should bear its quantitative fruit well within the lifecycles of successive baselines. With a quick enough process, the creation of architecture trade results can emerge comfortably ahead of any design crystallization. More breathing room allows time for more creative searches into non-standard tradespace regimes, increasing the probability of discovering unexpected yet promising design directions.

Steps in the Process

The MTT is geared to dramatically expedite the process of performing broad, shallow tradespace investigation. Any successful strategy to accomplish this objective must minimize the duration of each necessary step in the process:

- Time to understand customer needs
The MTT uses a real-time trade menu builder that can serve as feedback during a customer interview, eliminating trade ambiguity, and enforcing the rigor required for an effective quantitative approach.
- Time to architect a trade tool
Since the MTT is already built for maximum trade flexibility and is inherently trade-friendly, no time is wasted constructing trade architecting software from scratch. Periodic updates to the architecture for new desired functionality happen outside the customer cycle as pre-planned product improvements.
- Time to gather data and relationships
If the customer needs fall within the domain of existing models, this step is bypassed. Otherwise, the MTT framework accepts the best available information gathered during the allotted time.
- Time to enter data and relationships into the trade tool
The structured layout and transparency of the MTT has been developed to speed this process as much as possible.
- Time to troubleshoot
The MTT incorporates software curing common and semi-common data entry mistakes, such as trade consistency. It also forces resolution of all identified errors before they propagate during

tradespace computation.

- Time to compute populated tradespace
This varies based on number of trades, options per trade, and complexity of algorithms. When tradespaces become too large to exhaustively compute all permutations within a reasonable timeframe, advanced search algorithms need to be applied. These techniques are well known.
- Time to extract meaningful results
An oft overlooked step. This is composed of two functions. The first is to make sense of the large amount of data generated to arrive at meaningful conclusions. The second is to convert that information into data products usable by downstream customers, such as auto-generated equipment lists for promising architectures. Several tools have already been built/incorporated and several more are planned.

2. DATA SOURCES

One of the keys to MTT's flexibility lies in its data-driven format. This allows it to accept data of various types that can be rapidly configured to address a particular high-level tradespace need. It does not suffer the inherent limitations of pre-assembled system models that are useless outside their fixed assumptions. Indeed, it would be impossible to pre-assemble any model or collection of models that anticipate all potential directions of tradespace interest that the MTT might be tasked to explore. The MTT is mostly relieved of this burden through its role as repository of system and trade information that originates elsewhere (outside the tool). Therefore, the system engineer/analyst operating the MTT can draw from whatever data source is available and use the tool to facilitate data assembly into a trade-friendly format for analysis. When simple models offering reasonable sizing relationships appropriate to the demands of a particular tradespace are available, they can be used. Otherwise, the tool can accept trade impact estimates from experts or teams and use analogies from databases.

Figure 3 shows some various data sources along with timeframes and levels of fidelity typically encountered. The higher the level of fidelity of input data, the longer it usually takes to arrive. The capabilities of the MTT are tailored for the low fidelity/quick turnaround end of the spectrum. However, it has the ability to accept and structure data supporting system trades of higher complexity and increased nuance, at the expense of process rapidity. This trait is quite useful to project teams or other customers who desire a progression of tradespace analysis focus over time, from broad, shallow, and fast to deep, narrow, and slow. Of course, at the higher fidelity end of the spectrum, the MTT loses its advantages compared to other methods of tradespace examination, and becomes completely

inappropriate when dealing with simulation-level analysis. Nevertheless, by accommodating data of varying fidelity and system interactions of varying complexity, the MTT delays its obsolescence for those customers whose projects have matured yet who are reluctant to forego the quantitative trade benefits to which they have become accustomed.

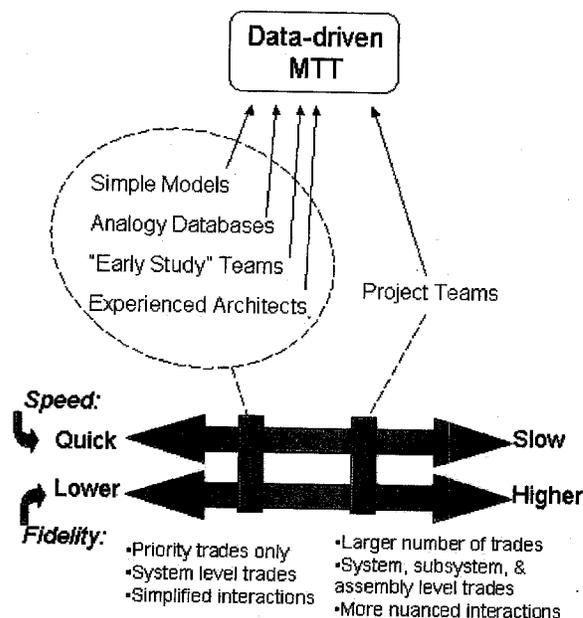


Figure 3: Data Sources

3. TERMINOLOGY

The MTT uses JAVA as its implementation language and XML as the means to store the description of a "Project". The definition of a "Project" and other framework items are provided in the following list of important terminology:

- Project – The root level entry of the MTT. Each XML file contains the data for a project, which typically correlates to a particular mission being investigated. A project comprises one or more Profiles.
- Profile – A profile represents a distinct approach to a mission. For example, a human mission to Mars might utilize a docked collection of vehicles lofted on a single heavy lift launcher or a time-staggered approach with the cargo sent to Mars months before the Earth departure of a crewed spacecraft. Profiles consist of one or more phases, and are meant to be tradeable.
- Phase – Phases are used to introduce sequence dependency into the MTT. They also contain

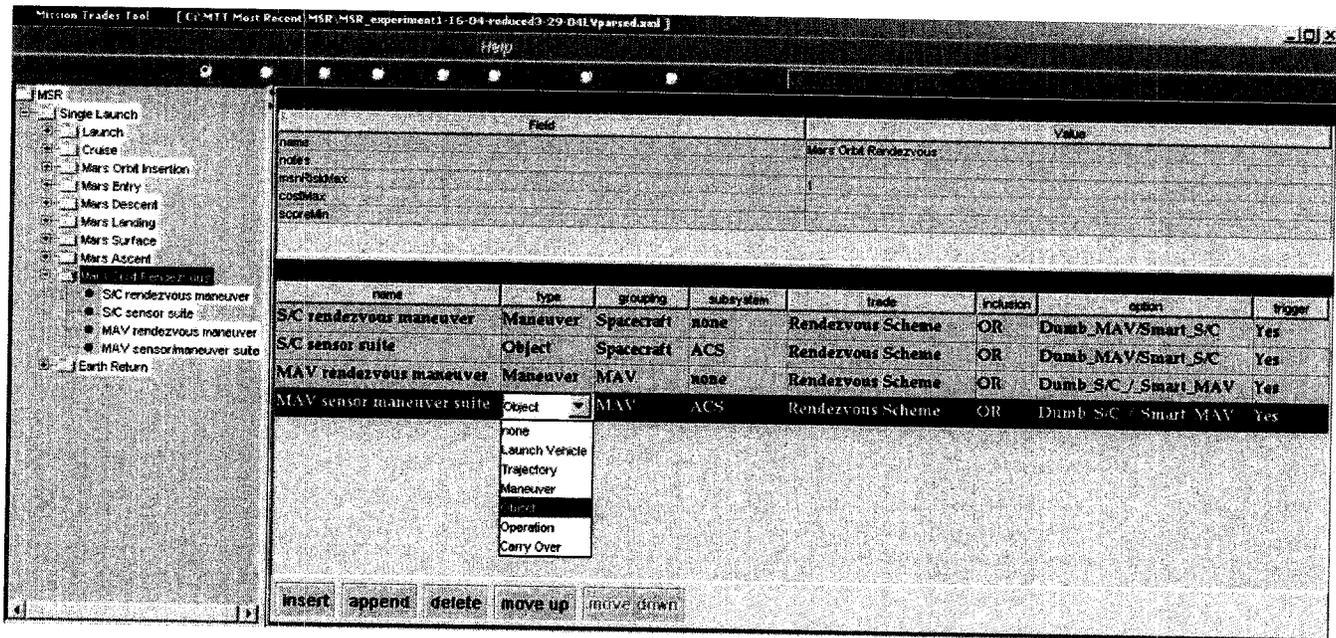


Figure 4: User Interface (Populated Example)

collections of elements that logically or conveniently are relevant within them. Phases offer the basic structure in which to define trades.

- **Element** - Elements are the generic building blocks of MTT. For example, an element can be a trajectory, a subsystem, a rover, a maneuver, etc. Elements have properties that distinguish their type, hierarchy and trade grouping relations, related logic conditionals, and attachment status. Elements also have attributes, which describe their features. Elements are tradeable entities.
- **Attributes** - Attributes describe elements. Some attributes are so common that they apply to all elements. These include placeholders for mass, cost, types of mission return, and modifiers for probability of success. Other attributes are automatically applied only to certain elements. Examples include delta V and specific impulse for maneuvers. The user can define an unlimited number of attributes to describe any element as the need arises. Attribute values are tradeable and calculable entities.

4. USER INTERFACE

The system engineer/analyst populating the MTT with data uses a dynamic graphical user interface, a populated example of which is shown in Figure 4. On the far left is a tree structure showing the project (a generic Mars Sample Return), the profile (Single Launch), and closed folders for

all but one of the mission phases. The selected mission phase (Mars Orbit Rendezvous) shows its elements, of which some selected properties are displayed in the elements editor comprising the main field inhabiting the center and bottom right of the figure. For this example, there are four elements bookkept in this phase, all of which are part of a single trade named "Rendezvous Scheme." The top two elements of the trade (a maneuver and sensor suite grouped with the spacecraft) are part of the "Dumb_MAV/Smart_S/C" option, whereas the bottom two elements (related to the Mars Ascent Vehicle (MAV)) belong to the other option. The "Rendezvous Scheme" trade is triggered by a "Yes" option (which appears in Figure 4 as part of the "Orbital Rendezvous" trade.) This means that a selection for an orbital rendezvous (as opposed to a direct earth return) is a prerequisite for the choice between rendezvous schemes. Conditionals for triggers can use a wide variety of logical operators. Also of note in Figure 4 is the pull-down menu that displays some commonly used element "types." All element property fields either accept direct data entry or provide smart pull-down lists that can automatically reconfigure depending on certain criteria. The field above the elements editor displays unique attributes for elements when they are selected in the tree structure. In Figure 4, since the "Mars Orbit Rendezvous" phase is selected, the field offers space for general phase information.

5. VIEWS AND FEATURES

Before exploring the tradespace, it is necessary to define the options and logic that comprise it. This step, although quite straightforward and less glamorous than producing

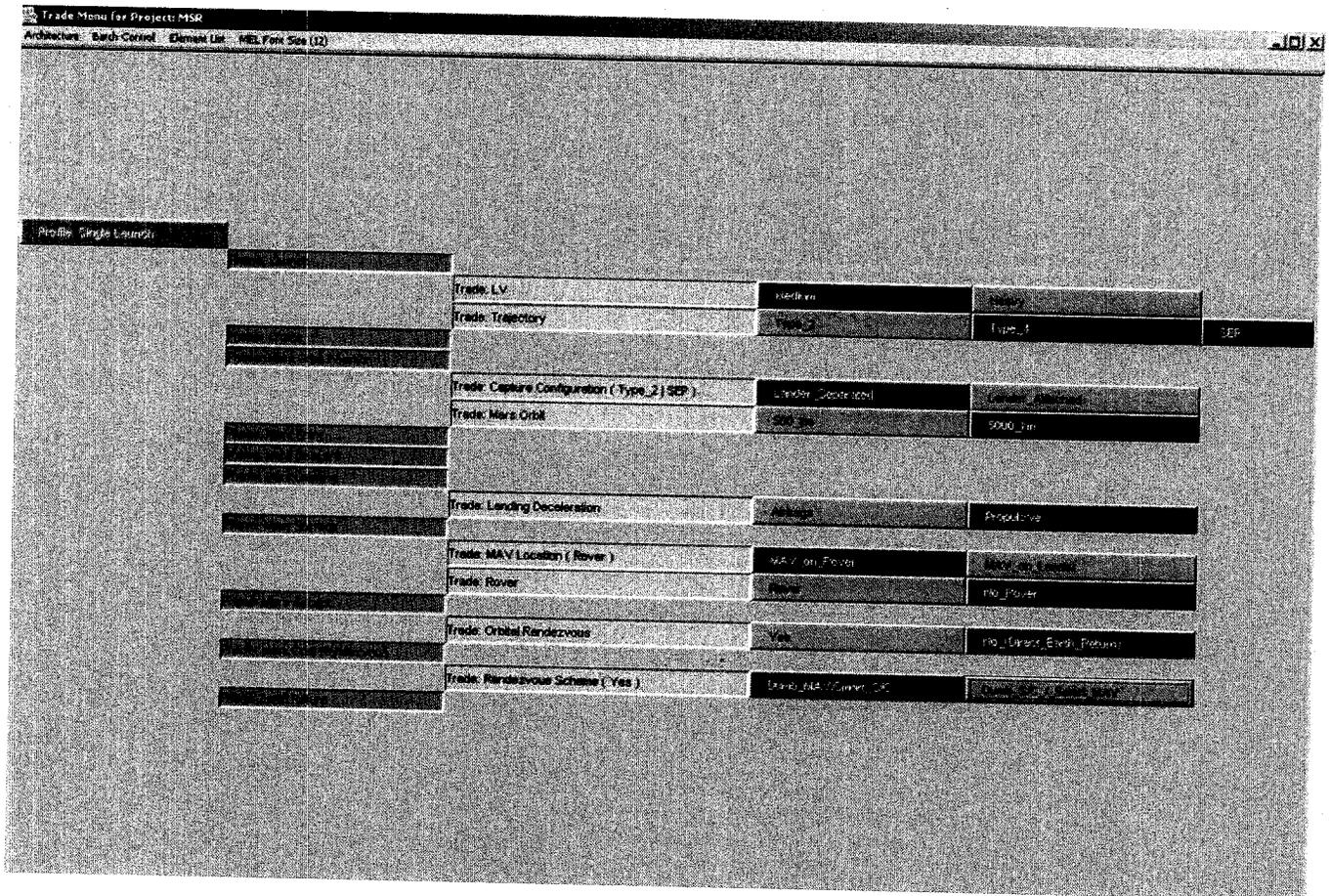


Figure 5: Trade Menu

populated tradespace plots, is among the most important functions the MTT facilitates. An astonishingly high likelihood exists that the trade options and related logic a project's leadership might want to investigate are inadequately communicated to those either performing the trade studies or supplying supporting data. A simple, standard way of capturing the trade options and displaying them goes a long way toward reducing confusion and keeping the engineering staff synchronized. After populating the user interface with just trade option definitions and trigger logic, a trade menu can be automatically created. Such a menu is shown in Figure 5 for a generic Mars Sample Return example mission. The trade menu format, (as opposed to a trade tree format) efficiently shows a surprising amount of information in a condensed, easily readable graphic that does not suffer from space wasting branch repetition. Because the MTT provides structure for rapid trade option creation, and since no underlying data such as mass, cost, or hierarchy information is necessary to create a trade menu, producing the menu, modifying it, and displaying it in real time to elicit feedback from a live customer can be a painless process. The agreed-upon trade menu then becomes the electronic skeleton around which to coalesce supporting system data when it becomes available. With the underlying data in place, the

trade menu becomes a clickable user interface with two modes:

Mode 1: Manual. In this mode the user can pick and choose various trade options as desired. Triggered trades become selectable once their prerequisites are satisfied by prior selections or combinations of selections. Selected trade options appear orange, while deselected ones show up dark green. After completely defining an architecture by resolving all outstanding trade choices, the user can generate the computed architecture and plot it against a large variety of selectable measures of effectiveness. At this point, the user may reference several standard windows that display the architecture's Mass Equipment List (MEL), segment hierarchy by phase, and propellant/maneuver information.

Mode 2: Batch. In this mode the computer performs an exhaustive calculation to generate all possible architectures. The resulting point cloud representing the feasible and infeasible architectures becomes available for tradespace analysis. Embedded tools facilitating trend analysis make reaching meaningful conclusions significantly easier for large architecture datasets. All views and output metrics for any architecture are immediately retrievable.

For ease of use, it is very important that the information implanted in the tool does not become shrouded in a “black box”, impenetrable to all interested parties except the user. Therefore, much care has been expended to enable data and structure transparency. The following Segment Hierarchy Display shown in Figure 6 is one of several standard example views that provide situational awareness. This is particularly important for relatively complicated missions like a sample return that may require numerous segments to separate and reattach in different combinations for various phases with all the attendant propellant bookkeeping demands of segment-specific maneuvers. The example shown highlights the hierarchy during a “Mars Descent” Phase.

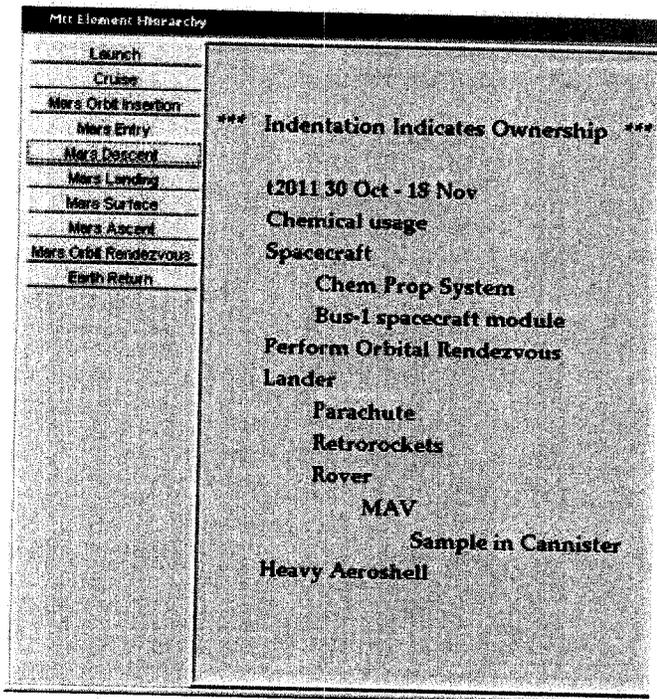


Figure 6: Segment Hierarchy Display

The following is a list of key features that MTT users can leverage:

- Automatic convergence of user-supplied simultaneous equations
- Unlimited attribute creation
- Embedded propellant bookkeeping
- Equation builder
- Cost wrapping and toggle-able cost estimating assistance

- Launch vehicle solver
- Selectable feasibility constraints
- Option filtering
- Automatic data consistency checking
- Seamless output file sharing to Excel and advanced tradespace visualization software.

6. CONCLUSIONS

The Mission Tradespace Tool offers its users a rapid, flexible method to define and explore a rough, high-level tradespace over a wide variety of potential mission types. Despite its unique capabilities, it remains dependent on a knowledgeable system engineer/analyst to exploit its relatively open framework in a meaningful manner. This is not necessarily a drawback, since it maximizes the relative strengths of computer frameworks and the unmatched problem definition capabilities of a skilled human. Attempting to remove the human from the loop is not something the MTT development community cares to tackle.

The algorithms that conduct much of the bookkeeping functions (e.g. propellant expenditure, hierarchy, equation convergence) have been rigorously verified over a range of assumptions. Early validation exercises have shown considerable promise for identification of relative system trends and the identification of rough feasibility/infeasibility. Absolute validation against higher-fidelity baselines with small error tolerances over wide tradespaces lies arguably outside the intended scope of rapid early feasibility exploration. Nevertheless, the issue of absolute validation remains a goal that will be approached through increased operational experience and the intelligent historical feedback mechanisms that a standardized trade capture framework allows over time.

Improvement in the end-to-end cycle time will be a target for a variety of MTT upgrades, both algorithmic, and procedural. Referencing a growing library of simple executable models and linking to increasingly relevant databases appears desirable. Yet even in its current form, the MTT has changed the landscape of expectations for early architectural design phases. For perhaps the first time ever, trade tool reconfiguration is no longer the speed limiting factor for continued quantitative explorations of widely varying space mission tradespaces.

REFERENCES

- [1] Hugh McManus, Daniel Hastings, and Joyce Warmkessel, "New Methods for Rapid Architecture Selection and Conceptual Design," *Journal of Spacecraft and Rockets*, Vol. 41, No. 1, 2004, pp. 10-19
- [2] Adam Ross, Daniel Hastings, Joyce Warmkessel, and Nathan Diller, "Multi-Attribute Tradespace Exploration as Front End for Effective Space System Design," *Journal of Spacecraft and Rockets*, Vol. 41, No. 1, 2004, pp. 20-28
- [3] André Girerd and Robert Shishko, "Project Trades Model for Complex Space Missions," *AIAA Space 2003 Conference Proceedings*, September 23-35, 2003.

BIOGRAPHY



André Girerd double-majored in Aerospace Engineering and History at the University of Virginia, and earned his Masters in Astronautical Engineering from MIT. In the nineties, he joined the operations staff for the Air Force's MSTI-3 satellite and the Iridium constellation. Currently, he works at the Jet Propulsion Laboratory as a space systems engineer for projects in their early phases, applying model-based design approaches to the Mars Telecom Orbiter and Prometheus 1.

ACKNOWLEDGMENTS

The author would like to thank the members of the Mission Tradespace Tool team at JPL who contributed their time and energy to development and testing: Marc Pomerantz, Joe Mrozinski, Gene Lee, Babak Cohanin, and Eric Wood.

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

Reference 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 of the Jet Propulsion Laboratory, California Institute of Technology.