X-TOS

16.89 Final Design Report
May 21, 2002

Acknowledgments

The authors would like to thank:

Dr. Joyce Warmkessel, Dr. Hugh McManus, and Professor Dan Hastings for instructing the class and guiding us through the process.

Nathan Diller, Adam Ross, and Satwik Seshasai for their valuable support, advice, and guidance throughout the course.

Dr. John Ballenthin (AFRL), Kevin Ray (AFRL), and The Aerospace Corporation for accepting to be our users and customers, and for all of their time and support.

Dr. Rick Fleeter, Ray Covert, and Dr. Myles Walton for their lectures on Space Systems.

Dr. Joel C. Sercel (Laboratory for Spacecraft and Mission Design, Caltech) for allowing us to use DrawCraft, a spacecraft configuration tool.

Col. John Keesee, Dr. Annalisa Weigel and Julie Wertz for their valuable assistance in their areas of expertise.

Fred Donovan for his computer assistance.

Authors:

Mirna Daouk
Jason Derleth
Kevin Duda
Bobak Ferdowsi
Deborah Howell
Geoff Huntington
George Leussis
Stephen Long
Christopher Roberts
Nirav Shah
Tim Spaulding
Dave Stagney
Ludovic Talon

Editors:
Stephen Long, Nirav Shah

Document Manager:
Jason Derleth

Table of Contents

List of Figures

List of Tables

1 Executive Summary

The purpose of this document is to summarize and describe the process completed and results obtained by MIT's 16.89 class during the spring semester of 2002. The class completed an architectural study for and a preliminary design of a satellite to measure specific regions of the upper atmosphere. The primary stakeholders and participants of the project are: 16.89 Students, faculty and staff, and AFRL.

Motivation for completion of this project is twofold: First, from a user driven perspective (AFRL), the design of a space system would provide valuable data for evaluation and short term forecasting of atmospheric density behavior, thus allowing improved models for satellite reentry location. Secondly, from a pedagogical standpoint (student and faculty), the class serves as a testing ground for the evaluation of new and innovative design processes while teaching and learning the fundamentals of space system design.

The objective of the design process is development and justification of a recommended space system architecture to complete the mission, and a preliminary design of this space system. The objective of the faculty is to ensure that the completed design process is adequately critiqued and assessed, as well as to ensure that 16.89 students are versed in the process and the fundamentals of systems design of a space-based architecture. These ideas are captured by the class mission statement:

> ***Design a conceptual space-based space system to characterize the upper atmosphere, with specific emphasis on the thermosphere and ionosphere. Building upon lessons learned from A-TOS and B-TOS, develop an architecture for the space system by March 22, 2002; building upon lessons learned from C-TOS, complete a preliminary design of this architecture by May 15th, and link this preliminary design back to the process used for the architectural study. Learn about engineering design process and space systems.***

Six primary measurements are to be completed by the actual space system:
1. Satellite drag
2. Neutral density
3. Neutral winds
4. Neutral composition
5. Ion composition
6. Temperature

To accomplish these measurements, a set of three instruments had previously been chosen by the scientific user of the system, so the class categorized the use of these instruments as one of the design constraints.

To design such a system, an innovative design process was utilized to develop a series of space system architectures that complete mission objectives, while calculating the utility, or relative value of each, as weighed against cost. This design process eliminates the potential to miss optimal solution options by focusing on a point design, but rather gives to the primary user a host of choices that can be compared based on their relative value.

Upon completion of the architectural study, a baseline architecture was selected to be used as an initial beginning for completing the preliminary design. Again, an innovative design process was utilized to develop this preliminary design. The class used Integrated Concurrent Engineering to complete a preliminary design. The main parameters that drove the utility of this design were captured, utility was computed and compared to the expected utility from the architectural study.

Upon completion of the preliminary design, a sensitivity study was completed, this report was written and a final presentation given. Within this document will be found the particular challenges that were found by the class throughout the processes described, as well as: assumptions made; lessons learned; a description of the code used to calculate utility and cost for the architectural study; justification for the design chosen; a description of the method and process used for the Concurrent Engineering sessions; a description of the preliminary design; and, finally, overall lessons learned, observations made, and suggestions for future design classes.

2 Introduction

During the spring semester of 2002, the Space Systems Engineering (16.89) Class addressed the design of a space system—the X-Terrestrial Observing Spacecraft (XTOS)—that will provide data for the improvement of existing atmospheric drag models. In its approach to the space system design, the class employed a unique and ground-breaking process termed MATE-CON (Multi-Attribute Tradespace Exploration with Concurrent Engineering). The purpose of this document is to capture the motivations, key highlights, and lessons learned throughout that design process and to discuss the space system produced. The primary participants in this project are the 16.89 students and faculty, with the Air Force Research Laboratory (AFRL) serving as the primary stakeholder. In addition, the Space Policy and Architecture Research Consortium (SSPARC) has a rooted interest in implementing the MATE-CON process in a realistic space system design environment.

2.1 Motivation

The motivation for this project is twofold. First, from a user-driven perspective (AFRL), the realization of the space system would provide valuable data for evaluation and short term forecasting of atmospheric density, thereby improving orbital debris tracking and satellite reentry location prediction. Secondly, from a pedagogical standpoint (students and faculty), the class serves as a testing ground for the evaluation of new and innovative design process that simultaneously aids the students in learning the fundamentals of space system design.

2.2 Objectives

There are two main objectives that drive the direction of the class. First, the students are to develop and recommend a system architecture with top-level requirements, based on stakeholder constraints and user needs and preferences. Second, the students must provide a preliminary design of this architecture. Functional, design, and operational requirements are established for the detailed design of both the ground and space segments, as well as a preliminary design for the space component.

2.2.1 Mission Statement

The mission statement for the X-TOS project is determined through class and faculty discussion. The primary goals of the mission statement are to articulate:
what the project is about,
why the project should be undertaken, and
how the project will be done.
The X-TOS mission statement is then as follows:
To design a conceptual space system to characterize the properties of the upper atmosphere, with specific emphasis on the thermosphere and ionosphere. By building upon lessons learned from A-TOS and B-TOS, develop the design tradespace and select an architecture for the space system. The class will then complete a preliminary design of this architecture using concurrent engineering methods and incorporate a feedback loop from the preliminary design to the tradespace exploration process used for the architectural study. Learn about the engineering design process and space systems engineering.

2.2.2 Assessment Methods

The objective of the faculty is to ensure that each point in the design process is adequately critiqued and assessed, as well as to ensure that the 16.89 students are versed in the fundamentals and trades of space systems design. Some assessment is conducted through the class deliverables, which include the spacecraft preliminary design, the final report document, and two formal presentations. These deliverables constitute major milestones in the progress of the class. Table 2-1 summarizes the class milestones and the point in the semester that they occur.

Table 2-1: X-TOS Milestone Dates

Deliverable	**Date**	**Purpose**
Midterm Architecture Presentation	3/20/2002	Presentation and review of architectural design
Final Presentation	5/13/2002	Presentation and review of architectural and preliminary design(s)
Final Report	5/20/2002	Submission of the final report document

In addition to the deliverables mentioned above, the class produced reusable sets of code for both the tradespace exploration and the preliminary design portion of the class. These code sets are considered deliverables as future classes may reuse them.

2.2.3 Class Value Proposition

At the outset of the semester, the faculty poses the following questions to the class to garner an understanding of what the class is most interested in:
What do you want from the class?
What do you expect to contribute to class?
Level of effort
Special interests
Special expertise
The role a particular student plays in each segment of the design process is somewhat determined by the special interests and expertise he/she indicates at the beginning of class.

2.3 Approach

The basic approach to the space system design begins with an understanding of the mission at hand and the development of the framework that will be used to explore the tradespace. After a thorough understanding of the mission is achieved, the architectural study is approached using the MATE method. This method, first developed by Adam Ross and Nathan Diller under the MIT Space Systems Policy and Architecture Research Consortium, uses scientific-user and customer values, as quantified by multi-attribute utility theory, as output metrics. These metrics guide the architecture selection process toward more efficient systems by incorporating decision-makers as part of the early conceptual design process. Throughout the approach, some compromises must be made concerning the fidelity of the design versus the priorities of the class so that the problem can be scoped to the time allotted (one semester).

2.3.1 X-TOS Mission Overview and Scope

The general purpose of the X-TOS mission is to characterize the density of the upper atmosphere using three instruments provided by the scientific user. These instruments are to be flown on a space vehicle and measure small perturbations in the satellite's orbit as well as directly measure the mass and constitution of the atmosphere in the satellite's path. The class determined after some discussion that the only method suitable for gathering the data requested by the AFRL is to fly the instruments. The instruments will make the following six measurements:

1. Satellite drag
2. Neutral density
3. Neutral winds
4. Neutral composition
5. Ion composition
6. Temperature

2.3.1.1 Motivation for Density Forecasting

The general purpose of the X-TOS mission is to characterize the density of the upper atmosphere for improved solar weather forecasting. The improvement of this forecasting ability will serve both the military and civilian communities. Militarily, it will provide data to permit more accurate modeling in three deficient areas – satellite tracking, close approach/collision avoidance, and orbiting body reentry prediction. From a civilian standpoint, improved reentry prediction will greatly enhance early warning capabilities for populated areas in the zone of impact.

Currently, the accuracy of the best atmospheric drag model at the AFRL is approximately 15% (total standard deviation)[1]. The current error-reduction process, called "now-casting," involves a correction of the model by feeding back data collected through real-time observation of satellites from the ground. However, this process improves the accuracy to only 5% (total standard deviation). Even worse, this process only corrects the instantaneous positions at the time of observation – it gives no improvement to prediction. In order to accurately predict future deviation of satellites from their predicted path, a better atmospheric drag model is needed. The goal of this mission is to produce an atmospheric model that is accurate to within 5% standard deviation.

2.3.2 Architectural Notional Flow

MATE differs from other design processes in that it allows a multitude of space system architectures to be evaluated against each other to find the best values in performance, cost, and user satisfaction. By spanning the tradespace at the conceptual level, this design process eliminates the potential to miss valuable architectural possibilities that may otherwise occur in the traditional requirements-driven "point design-and-modify" process. MATE exposes the decision makers to the entire spectrum of architectural options and reveals the relative value of each.

[1] From "Atmospheric Density Specification ADS AFRL-705**"** presentation by John Ballenthin

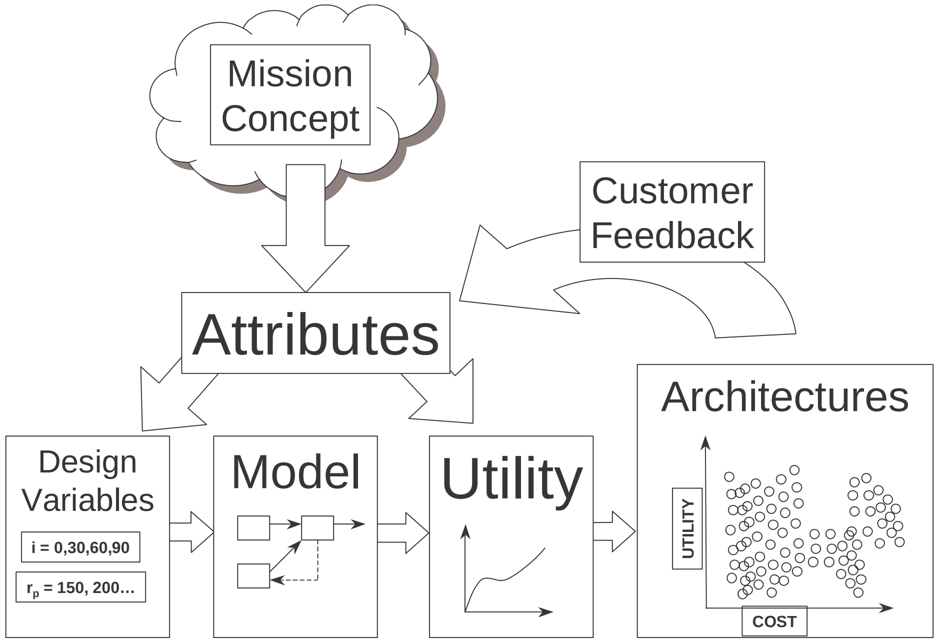


Figure 2–1: The MATE process

Figure 2–1 shows the notional flow of the architectural evaluation process for XTOS. A basic description of each of the different facets of this process follows:

Design Variables (Chapter 4) – the set of variables that determine each unique space system architecture. For XTOS, these variables include orbit apogee, orbit perigee, orbit inclination, spacecraft ΔV, communications architecture, antenna type, power type, and propulsion type. These include the *Constants Space / Constants Vector (Chapter 5 & 6*) – these are the different constants used in each of the modules. Some of these constants are typical astrodynamic and spacecraft subsystem constants, while others may be defined by the mission.

Model (Chapter 5 & Appendix E) – inputs a unique architecture defined by the design vector and passes out the resulting spacecraft and mission attributes.

Attributes (Chapter 3) – performance characteristics that offer increased (or decreased) value to the user based on their impact to the mission. For XTOS, the attributes were determined to be data life span, sample altitude, diversity of latitudes contained in the data set, time spent in the equatorial region, and latency.

Utility Function (Chapters 3 & 5) – relates the utility (based upon the customer's preferences) to each of the attributes.

Cost & Utility – comprises the final outputs of the model. Utility is typically plotted against cost to find the "optimal" architectures.

Architectures – the tradespace that emerges from running the models and is quite large in the case of X-TOS, with over 50,000 different architectures considered.

2.3.3 Integrated Concurrent Engineering notional flow

This year's class was augmented by the addition of Concurrent Engineering to the process to develop a preliminary design. Additionally, a new chair (which can roughly be regarded as a subsystem) dubbed MATE-CON was added to the Concurrent Engineering phase. The MATE-CON chair enabled real-time evaluation of a given iteration of the preliminary design in the same scientific user and customer value space that was used in evaluating architectural tradespace.

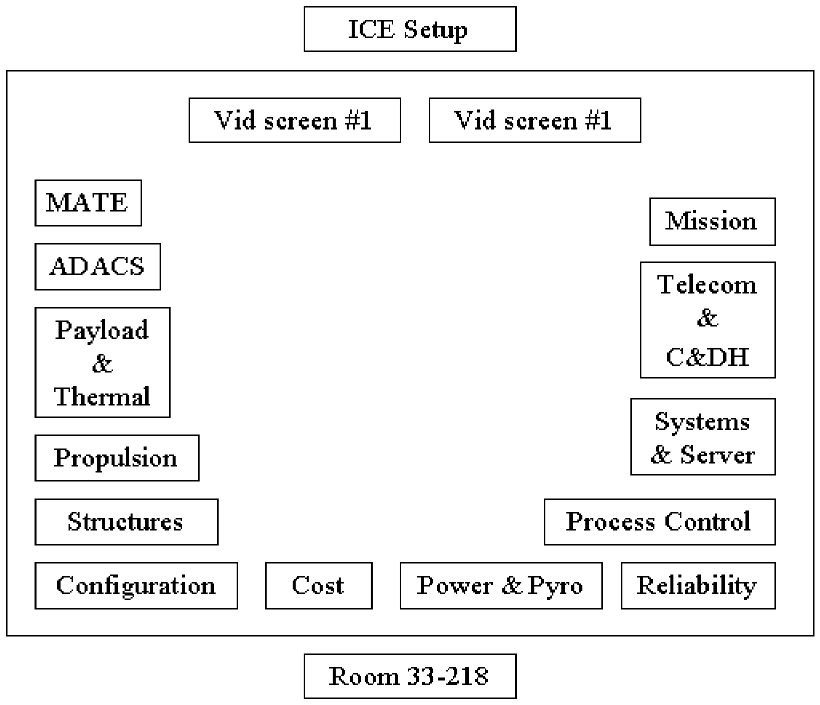


Figure 2–2: ICE Floorplan

After selecting an architecture from the architectural study, a preliminary design is completed. The Concurrent Engineering phase was completed in an Integrated Concurrent Engineering (ICE) environment in the Aero/Astro design lab:

In addition to having all of the engineers in the same room, a universal program, called ICEMaker (A product created by Caltech), is used. This program allows communication through the ubiquitous Microsoft program, Excel. Each chair is responsible for one or more Excel workbooks that, when taken together, instantiate the design of the satellite system.

A design is created by first creating or modifying an existing set of workbooks to reflect the unique challenges that the design is attempting to surmount. Each workbook includes an “inputs” sheet, an “outputs” sheet, and usually several pages of calculation sheets. These inputs and outputs are managed by the ICEMaker server, allowing the seamless transfer of hundreds if not thousands of variables between the calculation sheets that are managed by the subsystems chairs. X-TOS had nearly one thousand variables being passed between the sheets.

This variable passing allows the engineers to iterate through the design process and come to convergence rapidly. In the last day of design, the class had streamlined the process to such a level that an iteration towards convergence could be done every minute. At this rate, no more than an hour would be required for substantial changes to the design, assuming that no new code would need to be integrated into the subsystem workbooks.

2.4 Results

A series of architectures have been determined to be viable to satisfy the users' needs. MATE-CON was successfully implemented providing the scientific user with a focused tradespace of architectures to meet the desired architecture attributes. A preliminary design was completed, based upon the architectural study and tradespace exploration. Presentations, the Matlab code, Concurrent Engineering Excel sheets (based on the ICEMaker platform, a product of Caltech), and this document can be used by future classes or researchers for further application. The entire process facilitated student learning in the fields of engineering design process and space systems.

3 MULTI-ATTRIBUTE UTILITY THEORY

3.1 BACKGROUND AND THEORY

Multi-attribute utility theory (MATE) is a front-end process, designed to help engineers explore a tradespace of possible design architectures. Instead of beginning (like most system design processes) with a rigid set of requirements, MATE seeks to understand a user's preferences regarding specific qualities of the system. For example, where a typical process begins by setting a minimum acceptable lifetime, the MATE process seeks to understand the user's preference for a long or a short lifetime. By keeping designers away from focusing too quickly on a "point design," the MATE process allows for a wider tradespace, which can then be explored for possible best designs.

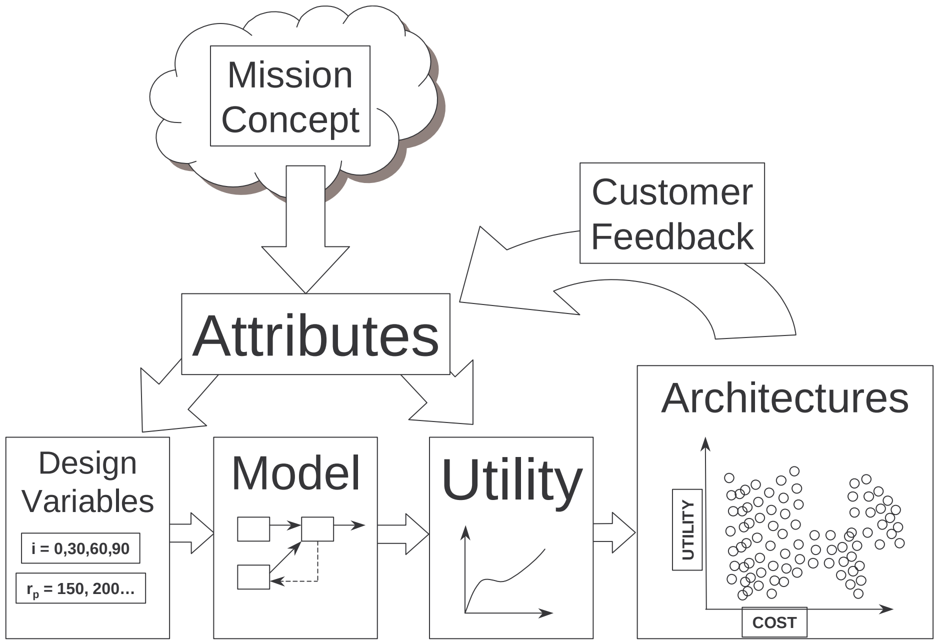


Figure 3–1: MATE Process

The keystone of the MATE process is Multi-Attribute Utility Analysis, which allows designers to get some numerical measure of the usefulness of the system. The process begins by breaking down the system's characteristics into a maximum of seven "attributes," for which there is a range of acceptable values. These attributes are agreed upon and commonly defined between the user and design team so there is a minimum amount of ambiguity. An example attribute might be phrased as "Length of time spanned by continuous data," and have a range from 6 months to 11 years.

The user is then asked, using a method called "Lottery Equivalent Probability" to give preferences balancing the level of the attributes against the chances of actually achieving them (For more detail on the Lottery Equivalent Method, see the appendix). These questions yield a single attribute utility function, relating the utility to the user (from 0 to 1) to the value of the attribute. Examples of these curves can be found the appendix. Finally, in order to get a measure of the relative importance of each of the attributes, the user completes a “corner point” interview, yielding a weighting factor for each attribute.

There are some special conditions on the attributes that must be met to fit the multi attribute utility theory. First of all, attributes must be monotonically increasing or decreasing. For instance, an attribute whose maximum utility was found in the middle its possible range would be unacceptable. Adjusting the range definitions to ensure that utility is maximized at either the beginning or end of the range can usually satisfy this condition. The attributes must also be limited to no more than seven. This comes about because of the “rule of seven,” loosely stating that human mind can consider only seven concepts simultaneously. Since the user is asked to consider all attributes simultaneously during the corner point interview, seven attributes is the practical limit. The other conditions require that the set of attributes be complete, operational, decomposable, non-redundant, and minimal. Meeting all of these conditions simultaneously can be quite a challenge, as we found out in the implementation.

Multi-attribute theory uses the utility functions and associated weighting values to construct a composite utility value for any given set of attribute values. This allows designers to imagine architectures, score their attributes, and see what sort of utility rating they receive according to the user’s preferences.

3.2 Attribute generation

For the X-TOS project, our first task was to seek out a person that who we could interview as the “user.” We initially held an information session with Dr. John Ballenthin, the designer of the instrument suite’s mass spectrometer. In this initial session, we probed the performance aspects of the instrument, as well as the use to which the data would be put. We also asked questions of Dr. Ballenthin regarding his preferences toward the system, which were: low perigee, high inclination, 12 month lifetime, .25 km altitude accuracy, .50 km position accuracy, and very accurate pointing. He further described his ideal data set to be: “instantaneous, global coverage.” This input allowed us to form an initial list of potential attributes. These attributes included spatial resolution, distribution of latitudes and longitudes in the data set, altitude, pointing accuracy, sample rate, and data lifespan.

This initial set of attributes began an iterative process where the Utility Team members, responsible for creating the list of attributes, took input and suggestions from the rest of the design team. After the first of many iterations, it became obvious that more detailed input from the user was necessary to explore more attributes, as well as to define the range values for those that were emerging in their final form.

For this detailed information, we elected to work with Mr. Kevin Ray. Kevin worked closely with the B-TOS team, and therefore had a thorough understanding of the MATE attribute definition process. We used Kevin as a liaison—we would ask him questions about possible attributes and ranges, and Kevin would answer, fueled with information from John. Discussions with Kevin focused around expanding John’s “requirements” into flexible range definitions. For instance, we took the 12 month lifetime number, expanding it into a range from 6 months to 11 years. This process of expanding requirements into an acceptable range for attribute definition allowed us to determine which qualities of the system would be attributes, and which would be handled as constraints. For instance, for the pointing requirements, Kevin made it clear that anything below the required level was unacceptable, while increments above the requirement added no value to the system. So, pointing accuracy became a constraint. Conversely, even though John stated that he wanted a 12 month lifetime, it became clear that lifetimes as short as 6 months and as long as 11 years had varying levels of value. This meant lifetime would be an attribute.

This iteration between the utility team, customer, and the rest of the design team continued until we condensed the attributes down to five:

Table 3-1: Final X-TOS Attributes

Attribute	**Units**	**Min**	**Max**
1) Data Life Span	[years]	0.5	11
2) Sample Altitude	[km]	150	1000
3) Diversity of Latitudes in Data Set	[degrees]	0	180
4) Time Spent in Equatorial Region	[hours/day]	0	24
5) Latency			
Scientific Mission	[hours]	1	120
Tech Demo Mission	[hours]	0.5	6

3.3 X-TOS Attribute Definitions

Data Life Span: Elapsed time between the first and last data points of the entire program, measured in years.

Sample Altitude: Height above standard sea-level reference of a particular data sample, measured in kilometers. (Data sample = a single measurement of all 3 instruments)

Diversity of Latitudes Contained in Data Set: The maximum absolute *change* in latitude contained in the data set, measured in degrees. The data set is defined as data taken between 150 – 1000 km.

Time Spent at the Equator: Time per day spent in the equatorial region defined as +/- 20 degrees off the equatorial. Measure in hours per day.

Latency: The maximum elapsed time between the collection of data and the start of transmission downlink to the communication network, measured in hours. This attribute does not incorporate delays to use.
Scientific Mission – Latency max and min for the AFRL model
Tech Demo Mission – Latency max and min for demonstration of now-casting capability.

These attributes have a few points of interest: First, they represent an attempt to capture the somewhat disparate preferences of the user. Though John Ballenthin originally said that he wanted a high inclination mission to cover the interesting regions of the atmosphere at high latitudes, Kevin refined this desire, noting that there were really two areas of interest—the high latitudes and the equatorial regions. Were this to be defined in a single attribute ("inclination" for instance), this would have violated the condition that all utility functions be monotonic—the curve would have been high at both ends, but low in the middle. So, instead, we formed two attributes to capture this preference. One captured the equatorial region (Time spent at the equator), while another captured the high latitude regions (Diversity of latitudes contained in the data set). It is interesting to note here that this pair of attribute definitions does not allow for mutual maximization, at least for the single satellite case). This is not a problem, but will keep the utility values for the single satellite case far below their theoretical maximum of 1. We validated this pair by noting that the utility maximizing pair of satellites would be one that flew in a polar orbit (maximizing the latitude attribute), and one that flew in an equatorial orbit. This is precisely what John mentioned that he would do if he had two satellites.

The second point of interest involves the altitude attribute, which presents a problem in definition. We wanted to consider both circular and elliptical orbits, with all variety of perigee and apogee combinations. However, there was no way to present these orbits to Kevin for evaluations of utility. Our solution instead was to interview Kevin regarding the utility of a single data point at a given altitude. This generated a curve of the utility of single altitude measurements. Using this curve, you can then break each possible orbit up into discrete, one-minute sections, assigning a utility value to each that is averaged over the entire orbit, yielding a utility value for the whole orbit. It was this orbit utility value that we used for further calculations.

The last point of interest is the latency attribute, which was split into two categories: latency for the science mission and latency for the tech-demo mission. This came out of a discussion with John Ballenthin, who mentioned that there was a desire to demonstrate a "now-casting" capability for ionospheric conditions. In order to accomplish this mission, much smaller latencies were required. This complicated matters somewhat during the utility interview phase, because Kevin had to do two separate multi attribute corner interviews—one using the science latency and one with the tech demo latency. In hindsight, this attribute was a poor choice, as design considerations made it easy to achieve the best latency.

3.4 Initial Interview (MIST software)

Having iterated several times and finally arrived at a complete list of attributes, we moved into the interview phase. As described above, the interviewing process uses the "Lottery Equivalent Probability" (see appendix for more information) to generate a utility curve for each attribute, as well as weighting values. In the past, conducting these interviews was a lengthy, arduous process, taking up many hours of both the interviewer and the interviewee's time. For our interview, we capitalized on the Multi-attribute Interview Software Tool (MIST), which is graduate work created by Satwik Seshasai. (For screenshots of MIST see the appendix).

The interviewing technique of "Lottery Equivalent Probability" centers around constructing plausible scenarios, allowing the user to decide between two alternatives. The challenge in constructing these scenarios is keeping the user focused on the model, instead of a satellite solution they may have in mind.

3.4.1 Data Life Span:

A ground station has developed the technology to accurately extract pertinent data for the AFRL model. This ground station will significantly increase data life span as compared to current systems. However, this new ground station has uncertain long-term funding. Your design team has studied the issue. They indicate that the current technology will give you a 50% chance of getting a XX data life span or 0.5 years. The new technology will give you a ## chance of getting a data life span of 11 years or a 1-## chance of getting 0.5 years.

3.4.2 Sample Altitude:

A commercial television provider has offered to place a sensor on its geo-synchronous satellite with a lookdown capability to extract pertinent data at 150 kilometers. However, there is a chance that the instrument will become misaligned due to launch vibrations. Your design team has studied the issue and determined that any misalignment will cause the sensor to extract data at 1000 km. You must decide between using this sensor, or traditional methods. The traditional methods will give you a 50% chance of getting data at XX km, or a 50% chance of getting data at 1000 km. The new sensor has a ## chance of extracting data at 150 km or a 1-## chance of extracting data at 1000 km.

3.4.3 Diversity Latitudes Contained Data Set:

A boat-based sensor capable of collecting pertinent data promises to offer a wide diversity of latitude. However, there is a chance that the boat will never leave port due an ongoing seamen strike. If you elect to use traditional methods there is a 50% chance that you will get XX degrees of diversity in latitude of your data, or a 50% chance that you will get 0 degrees diversity of latitude in your data. The boat-based sensor offers a ## chance of getting 180 degrees of diversity of your data or a 1-## chance of getting 0 degrees of diversity of your data

3.4.4 Time Spent in Equatorial Region:

New instruments capable of extracting pertinent data to the AFRL model have been installed on an equatorial ground station. Use of this ground station can get you equatorial data. However, there are many scientific users competing for sole use of these instruments. If you decide not to use this ground station in favor of standard measurement methods, you have a 50% chance of getting XX hours per day of equatorial data or a 50% chance or getting 0 hours per day. Using the new ground station you have a ## chance of getting 24 hours per day or 1-## chance of getting 0 hours per day.

3.4.5 Latency:

Latency is solely a function of communication capability with the ground via a satellite communication system. A new communication system is currently being assembled in space. Satellites are being added to complete the constellation and to provide an increased performance. The constellation is scheduled to be completed before the launch of your mission, although there is always some uncertainty about scheduling. You are studying whether you want to use the currently available communication satellites or this new constellation. Your design team has studied the issue. They indicate that the current satellite communications system will give you a 50% chance of getting a latency value of XX or 120 hours. The new satellite system will give you a ## chance of getting a latency value of 1 hour or a 1-## chance of getting a latency value of 120 hours. Which communication system would you use?

Compared to past iterations, the interview process ran very smoothly, allowing Kevin to complete the interview in the course of a few hours. The data from the interviews is plotted in the program, and output for further use. Below is a typical utility curve generated from the interview.

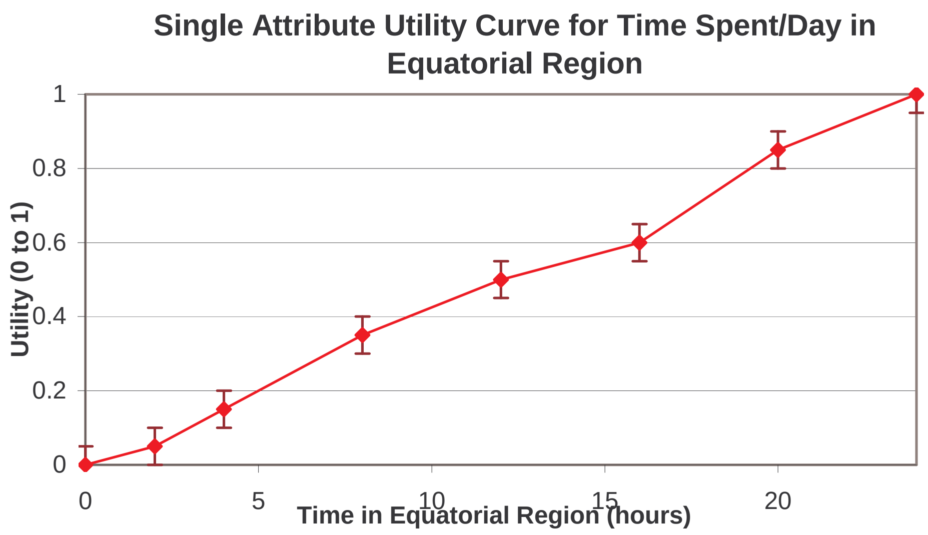


Figure 3–2: Single attrubute utility function for Time spent in Equatoral Region

This curve is relatively straightforward. The attribute value, bounded on from below by 0 hours per day and from above by 24 hours per day, is monotonically increasing across the range. In this instance, the preference seems to show little sensitivity to the attribute value—the relationship is linear. This is not necessarily so for all cases, as shown in the next graph:

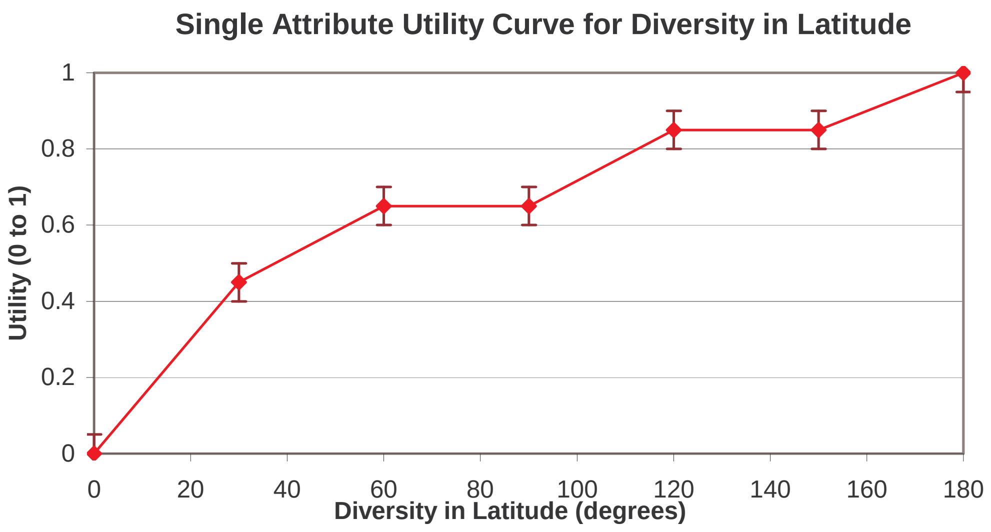


Figure 3–3: Single attrubute utility function for Diversity of Lattitude

In Figure 3–3 we see a more interesting case, as utility shows a more complex dependence on the diversity of latitude in the data set. The remainder of the utility curves can be found in the appendix.

MIST also demonstrates for us the validation of the key assumption of Utility Independence. This must be satisfied in order to ensure that the Multi-Attribute Utility Theory mathematical framework can be applied. To show Utility Independence, MIST found Kevin's indifference points (the points at which he was ambivalent between a lottery and a certainty) for each attribute in two different cases—one where all the other attributes are maximized, and one where all the other attributes are minimized. If his preferences toward the attributes are independent, the utilities should be the same in either case. The results are shown in Table 3-2.

Table 3-2: XTOS Utility Independence Results

	All others maximized	All others minimized
Data Life Span	0.375	0.375
Sample Altitude	0.325	0.325
Diversity of Latitudes	0.325	0.325
Time Spent at the Equator	0.125	0.075
Latency (scientific)	0.425	0.425

We see here that Time Spent at the Equator is the only attribute that seems to challenge the utility independence assumption. It is difficult to say what might have caused this apparent utility dependence. However, since the difference in value is not large, it may very well be simply a consequence of human fallibility—it is sometimes difficult for the interviewee to remain entirely consistent over the course of the interview.

The final function of the MIST software was to generate the weighting values for each attribute (called "k" values here). These values are scaled to 1.0, and show the relative importance of each attribute to the overall utility. For this interview, there are as follows:

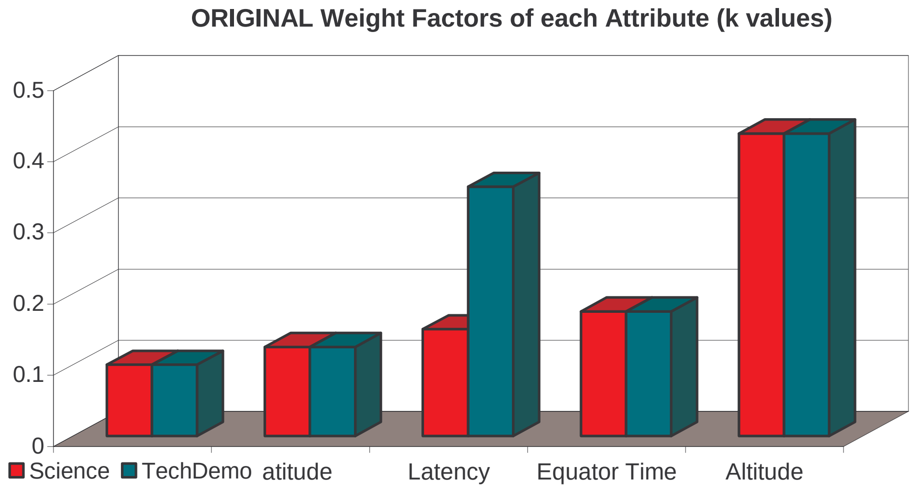


Figure 3–4: Weight factors of each attribute

3.5 VALIDATION INTERVIEW

One of the significant selling points for MATE-CON is the capacity for user involvement throughout the entire design process. This improved communication not only facilitates clearer insights to the user’s needs, but also expedites the design process as a whole. The utility team contacted Dr. Kevin Ray for a validation interview to confirm results from the first interview, and to gather any changes in user preferences. This second interview was completed April 25, 2002, in the midst of ICEMAKER design iterations. The new user preferences mentioned by Kevin Ray were effortlessly streamlined into the ICE process, and produced new designs within the day. It is important to note that for comparison purposes, both the old and updated user preferences were used to calculate utility. For the specific changes in the user preferences, please see Section 9.1 for more details.

4 Design Vector

The design vector[2] is a finite set of specific parameters that sufficiently define an architecture to compute the architecture's utility and cost. It is this set of parameters that are traded in creating the utility-cost tradespace. Each component of the design vector must be a specific quantity either chosen from a continuum (e.g. orbital inclination) or from a list of discrete choices (e.g. solar panels or fuel cells). For the MATE process to produce useful results, the design vector must satisfy two competing objective. First it must provide for a broad enough space of possible architectures so as not to eliminate possible high utility solutions that may not be intuitive to the designer. Second, it must be constrained sufficiently so as to allow rapid tradespace generation and quick turn-around of the architectural consequences of the decision makers expressed preferences. The design vector used in X-TOS tried to satisfy both of these objectives.

4.1 Design Vector Definition

The definition of the design vector begins with the consideration of user specified attributes (see Chapter 3). Since these attributes define user utility, and the objective of the designer is to maximize that utility it follows that the designer would choose a set of design variables that have a high degree of leverage in changing the values of these attributes.

In the case of the X-TOS attributes, two key groups of variables emerged: the orbit(s) in which data would be taken, and the spacecraft(s) taking that data. Specifically, the altitude, diversity of latitudes and time in equatorial region attributes are fully determined by the mission orbit, while the data lifespan and latency are functions of both the orbit and spacecraft.

4.1.1 Orbit Related Design Variables

Several different orbital strategies were considered. These ranged from simple circular orbit to decaying spiral trajectories. Some of the options considered are listed in Figure 4–1.

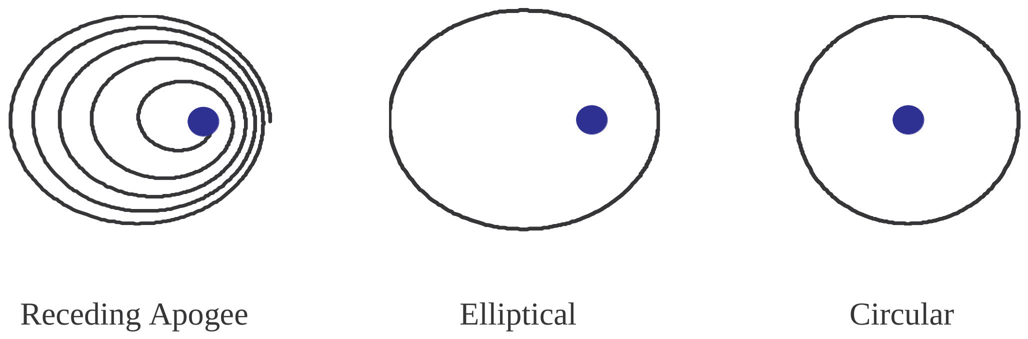


Figure 4–1: Orbital Choice

This set of options demonstrates the competing objectives in defining the design vector. The designer may wish to use the receding apogee strategy since it would appear to increase both the

[2] The reader is strongly encouraged to become familiar with Chapter 3 prior to reading this chapter.

data lifespan and altitude attributes (long life from being at a high altitude early in life, low altitudes towards the end of life as the orbit decays). When the class attempted to use this orbital strategy, it was quickly discovered that such a strategy is prohibitively difficult to model. Detailed information about the geometry of the spacecraft and environment in which it orbits is needed to even make a guess at how the orbit will decay. In fact, one of the objectives of the X-TOS mission is to provide data to improve models for precisely such a decaying orbit. The circular orbit on the other hand is computationally easier to handle. However, it limits the tradespace too severely, forcing the designer to choose a low orbit with a short lifespan or a high orbit with a longer lifespan. The X-TOS team chose the compromise of an elliptical orbit. It has computational simplicity of the circular orbit (also being keplerian) as well as the diversity of altitudes provided by the receding apogee strategy. The intention here is not to prove that the elliptical orbit is an optimal strategy, rather that it represent one possible compromise between the competing objectives of tradespace completeness and computational ease while still providing leverage over the attributes.

Having a general strategy for defining the tradespace is not sufficient to form the design vector; the designer must parameterize the space of possibilities using a finite set of variables. In the case of X-TOS orbits, three parameters were chosen: the altitude of apogee, the altitude of perigee and the orbital inclination. Of course these three parameters are not sufficient to fully specify a keplerian orbit; rather a total six orbital elements are needed. The remaining elements are not included in the design vector since they either do not provide leverage in changing utility or there is an obvious utility maximizing choice. For example, since only the latitude and altitude (not the longitude) of a particular data point is of interest to the user, the right ascension of the ascending node is not included. On the other hand, since the altitude and latitude range attributes are taken independently (i.e. the user is not expressing preferences for combinations of altitudes and latitudes) one would immediately choose the argument of perigee to align the line of nodes with line of apsides. Such a selection maximizes the time in the equatorial region without affecting the other attributes. These remaining elements are constants in the MATE model.

Unlike the orbits, appropriate design variables used to describe the spacecraft are not readily apparent from the attributes. In general, the computational and modeling resources available will tend to reduce the scope of possible architectures. The X-TOS team decided to eliminate concepts such as tethers since sufficiently fast and accurate models of their behavior were not in hand and could not be constructed in the time allotted. After reducing the scope of possible satellites, to relatively small traditional designs using off the shelf technologies, key sub-system level trades were identified.

4.1.2 Quality Function Deployment

A quality function deployment (QFD) approach was used to help understand key sub-system trades that affect the attributes. The QFD is a matrix in which columns are indexed by ‘qualities’ (for MATE, the attributes), and the rows by ‘functions’ that affect these ‘qualities’ (the design variables). Each element of the matrix is given a value between 0 and 9 indicating the relative impact that a particular ‘function’ (design variable) has on changing a particular ‘quality’ (attribute). QFD is used in choosing design variables since it both indicate the need for additional design variables as well the possibility the some variable may not be needed.[3] A small row sum indicates that the design variable indexing that row has a relatively small impact on the attributes and may be considered for elimination.[4] A small column sum indicates that the current set of design variables does not provide high leverage for that attribute and therefore additional design variables need to be defined. For X-TOS, a sample QFD is provided in Figure 4–2.

		Range	0.5 - 11	150 - 1000	0 - 180	0 - 24	1 - 120		0 - 200 M$	
		Units	Years	Km	Degrees	Hours (per day)	Hours		USD (2002)	
		Attributes	Data Life Span (Per Satellite)	Sample Altitude	Diversity of Latitudes contained in the Data Set	Time Spent in Equatorial Region	Latency	Total	Cost	TOTAL WITH COST
			1	2	3	4	5		6	
Variables	**Units**	**Constraints**								
Perigee Altitude	m	150 < hp < 350	9	9	0	0	3	21	9	30
Apogee Altitude	m	150 < ha < 1500	9	9	0	6	3	27	9	36
Inclination	Degrees	0 < i < 90	0	0	9	9	3	21	6	27
delta-V	m/s	0 < mass < 500	9	0	0	0	0	9	9	27
Comm System Type	-	AFSCN or TDRSS	0	0	0	0	9	9	3	12
Antenna Gain	-	Low or High	0	0	0	0	9	9	6	15
Propulsion Type	-	Chemical or Hall	6	0	0	0	0	6	6	12
Power System Type	-	Solar or Fuel Cells	6	0	0	0	6	12	6	18
Mission Scenario	-	-	9	9	9	9	3	39	9	48

Figure 4–2: QFD Matrix

From the QFD it is clear that the primary (the 9's), secondary (6's) and in some cases tertiary (3's) drivers have been identified for each attribute and that each design variable is a primary and/or secondary driver for one or more of the attributes.[5] A brief discussion follows of each of the non-orbit related design variables.

4.1.3 Summary of Spacecraft Related Design Variables

Total Delta-V capability: Lifetime is driven by this design variable since it gives the total amount of energy that can be expected over the life to counteract impulse from environmental and orbital perturbations. When this capability is exhausted the orbit can no longer be maintained and the operational life of the satellite ends.

[3] Note that computational efficiency is an objective of the design vector definition process. The more design variable used, the larger the tradespace, and consequently the longer it will take to compute utilities and costs. If certain potential design variables have an impact in certain parts of the tradespace but not in the space as a whole, one should first run the broader, simpler analysis without these variables, and then add the additional variables if the region that they affect is of interest. One can then compensate for the additional computing time required by the new variable by limiting the scope of trades done in the other variables to the interesting region.

[4] This does not necessarily mean that it will be eliminated for it may be the dominant driver for one attribute but not affect any others.

[5] This does not imply that *all* drivers have been listed. Indeed the vehicle's coefficient of drag is a key driver of the lifetime attribute. In this region of flow (hypersonic-particulate) the coefficient of drag is largely independent of vehicle shape and can be approximated as being near 2. As such it was constant in the model.

Communication System Type: The latency attribute is driven not only by the communication hardware on-board, but also by where the signal is being sent. Two choices were available for this mission, the Air Force's AFSCN and NASA's TDRSS.

Antenna Gain: The choice of antenna gain can significantly impact the latency attribute. For example, a high gain antenna must be steered and accurately pointed contributing latency. The two antenna choices also present different power requirements.

Propulsion Type: Given that this spacecraft will fly in a low orbit and will therefore need to expend a great deal of delta-V to counteract drag forces, having a choice a of propulsion systems is important. The trade-off represented by the two options of chemical and electric (Hall) propulsion is high-thrust at low efficiency vs. low-thrust at high efficiency.

Power System Type: The two propulsion systems represent vastly different power requirements. Since solar wings are not practical in the high drag environment these spacecraft will encounter, an alternative power source to body-mounted solar arrays, fuel cells, was considered.

The final design variable listed in the QFD is Mission Scenario. This variable is different from the others in that it is not related to a particular spacecraft, but rather deals with combining multiple spacecraft-orbit combinations into a multi satellite mission. Since there was no expressed requirement for data to be taken simultaneously at different points, each spacecraft in a multi-satellite mission can be taken as independent (i.e. their individual attribute values are not dependent on other spacecraft present). Several spacecraft can be designed independently and there attributes combined to form an overall mission utility. For example, if two spacecraft are launched in parallel the mission lifespan will be the greater of the individual spacecraft lifetimes. Three scenarios were considered in X-TOS: launch a single satellite, launch two satellites in parallel and launch two satellites in series.

4.2 Sampling Levels for Continuous Design Variables

The final step in defining the design vector is to choose at which discrete levels to sample the continuous design variables. The designer needs to choose a sufficient diversity of levels to ensure coverage of the tradespace, yet balance that choice with the additional computational expense of more levels. Often the number of combinations of design variables grows geometrically with the number of levels per design variable. The key is to use the attributes and utility functions to help define interesting areas of the trade space. For example in X-TOS, the levels orbital parameters were chosen to ensure breadth in inclination and a preference for low altitudes. There is some degree of art to this choice since one does not want to eliminate high utility areas of the tradespace. In X-TOS, the Total Delta-V design variable was capped at 1000 m/s (a cap that was thought to be conservative). During the detailed design phase (MATE-CON) it was discovered that values of in excess of 1200 m/s. were tenable. Table 4-1 summarizes the levels at which each design variable was sampled.

4.3 Iterative Nature of the Process

Though the process described here seems linear it is, in reality, quite iterative. At each step, as choices are made, the designer may discover that an error was made earlier in the process. An additional design variable could be in order, or, as models are developed, new computational constraints could emerge. In the end, the designer must communicate frequently with any modelers as well as with the user to ensure that the tradespace is sufficiently broad and sampled with enough resolution to capture the high utility architectures the user wishes to discover, but not be so difficult and time-consuming to model that the model results cannot be used as effective communication tool between the design team and the user adapting to changing (and hopefully more refined) user preferences.

Table 4-1: Design Variables and Levels at which they are sampled

Design Variable	**Levels**	**Justification**
Altitude of Apogee (km)	200:50:350[6]; 650:300:2000	Emphasis on low altitude in utility function, therefore sample at a higher rate at low altitudes
Altitude of Perigee (km)	150:50:350	Utility curve declines quite steeply between 150 and 350 km; will take a significant utility hit if spacecraft never flies below 350
Inclination (deg)	0; 30; 70; 90	Covers the possible range of inclinations
Total Delta-V (m/s)	200:100:1000	The low end of the range is a high average value for low earth orbit satellites. The high end is an estimate of the optimistic (on the large side) estimate delta V allowed before the spacecraft mass will no longer accommodate small and medium sized US launch vehicles.
Comm. Sys Type	AFSCN; TDRSS	Discrete choice of systems available
Antenna Gain	High; Low	Discrete choice of systems available
Propulsion Type	Chemical; Hall	See text above
Power Sys Type	Solar; Fuel	See text above
Mission Scenario	Single; 2 Series; 2 Parallel	More than two satellites is computationally prohibitive since the number of possible multi-spacecraft mission grows as N^k where k is number of spacecraft in the mission scenario and N is number of combinations of the other (spacecraft and orbit related) design variables.

[6] The notation *low* : *inc* : *high* means from *low* to *high* in steps of *inc*.

5 X-TOS Module Code Development

5.1 Overview

Once the design space has been fully defined, the relationship of each design variable to the overall spacecraft cost and utility needs to be mapped. The corresponding orbit, spacecraft, launch, and cost attributes for each design vector will ultimately determine the preferable architectures for this particular mission. A quick and efficient tool is needed to populate the tradespace with all possible combinations of satellite architectures and to perform a rapid cost and utility analysis of each combination.

5.2 Code Framework

Fortunately, the necessary tools already exist in the form of reusable Matlab modules created by previous 16.89 classes. Those modules were created to provide low fidelity calculations of the orbit characteristics (inclination, period, altitude, etc.), spacecraft characteristics (mass, power, size, etc.), the appropriate launch vehicles, cost, and utility for the tradespace of previous missions. Albeit some modification is necessary to adapt the modules to the current mission, the existence of previous versions greatly expedites the coding process.

5.2.1 The Matlab Environment

The modules are written in Matlab because it provides an integrated mathematical and visual programming environment in one easy to use package. The powerful numerical computation and graphing tools of Matlab allow the designers to quickly model the spacecraft subsystems, the space environment, and calculate the cost and utility of each design with little programming experience. Additionally, the ability of Matlab to interface with the Satellite Tool Kit (STK) provides complex orbit modeling capabilities that are otherwise be impossible to model for the average user.

5.2.2 Implementing the Database Structure

One significant improvement in the code framework this year is the implementation of global structures that can be saved as databases for later use. The software compiles a satellite database by running all possible design vector combinations through the models and collecting their respective orbital, physical, and launch characteristics. This database can then be fed into the cost and utility analysis modules to map the architectures to their utilities. This approach allows for the orbit, spacecraft, and launch modules to be run independently of the mission scenario, cost and utility modules and thereby greatly reduces the debugging and computational time. The utilization of this method also alleviates a significant bottleneck that occurred in previous versions of the code – the use of STK to generate orbit information in real time. By constructing a database of information for all possible orbits beforehand, the code simply pulls results from memory rather than having to access and run STK for each architecture. The only apparent drawback to this method is that the compilation of the satellite database requires large amounts of paging memory (the number of satellites can easily exceed tens of millions, depending on the size of the design vector) that can cause the computer system to hang. To circumvent this problem, the resolution of certain design variables can be reduced to limit the number of solutions. Additionally, the software automatically removes all satellites that cannot be launched from U.S. launch sites.

5.2.3 Scope and Resolution

As stated earlier, the XTOS modules exist to model the first-order effects of the design variables on the spacecraft architecture. The ultimate goal of this process is to determine the best architectures at the system level. To keep the number of solutions manageable, design variables with a continuous tradespace (like orbit apogee, orbit perigee, and spacecraft ΔV) must be sampled at rather large intervals. Ultimately, the reduced resolution of the design vector does not affect the results – the cost and utility relationships are still visible. In addition, concepts such as lifting bodies, tethers and swarms are not considered in the software because of scope and modeling concerns.

5.3 Software Computational Flow

The path followed in compiling databases and evaluating architectures is outlined below.

5.3.1 Satellite Database

The satellite database is created in order to map all of the possible variations on the design vector (with a certain resolution) to a specific satellite. The orbital module first varies the apogee altitude, the perigee altitude and the inclination, and then records each of these combinations. This forms the basis of the satellite database. Next the spacecraft module adds to the database by matching to each orbital variation parameters such as antenna gain, communication architecture, and propulsion type. If there is an opportunity to vary the spacecraft parameters on each satellite, it is done. For example, if a 300-kilometer circular equatorial orbit is able to communicate using either low or high gain antennas, then two variations on that orbit are produced – one with the low gain, and one with the high gain antenna. The spacecraft module is quite complex, and contains system sub-modules such as structures, power, and propulsion. Next, a launch vehicle is selected based on the minimal cost, the orbital parameters, and the physical parameters. This is then appended to each entry in the satellite database. All of the modules used are discussed in more detail in section 5.4. The existing database contains 9930 entries, and each entry includes orbital, physical and launch parameters. A sample entry can be seen in Table 5-1.

Table 5-1: Sample satellite database entry

data_error_flag	0
bad_launch_flag	0
id	1000
inclination	1.5708
alt_perigee	200000
alt_apogee	200000
comm_arch	0
total_delta_v	700
prop_type	0
power_type	1
ant_gain	1
period	5.31E+03
time_eq	19020
min_lat	-1.545
max_lat	1.562
delta_v	48.4051
alt_vector	[89x1 double]
bus_mass	175.5718

payload_mass	20.5
dry_mass	196.0718
prop_mass	8.9422
total_mass	205.014
latency	2.12E+04
lifetime	0.2726
lifetime_raw	0.2726
volume	2.5226
diameter	1.171
length	2.3421
max_avg_power	486.7966
max_peak_power	486.7966
tfu_cost	2.14E+07
lv_name	'Minotaur'
lv_cost	12500000
lv_site	Vandenberg or Kodiak'
arch_id	28

5.3.2 Mission Scenario Database

Since the number of all combinations in the satellite database is immense, the tradespace needs to be reduced so that the computation time for each mission scenario would be reasonable. Also, since there are an infinite number of different types of mission scenarios, only the most relevant, likely, and computationally inexpensive mission scenarios are chosen. The rules that are applied for the reduction are: 1) any satellites produced would be physically identical; 2) a maximum of two satellites would be orbiting at any particular time; 3) a maximum of two satellites would be produced; and 4) that only one satellite would be aboard each launch vehicle. There are three mission scenarios to which these rules are applied [Figure 5–1].

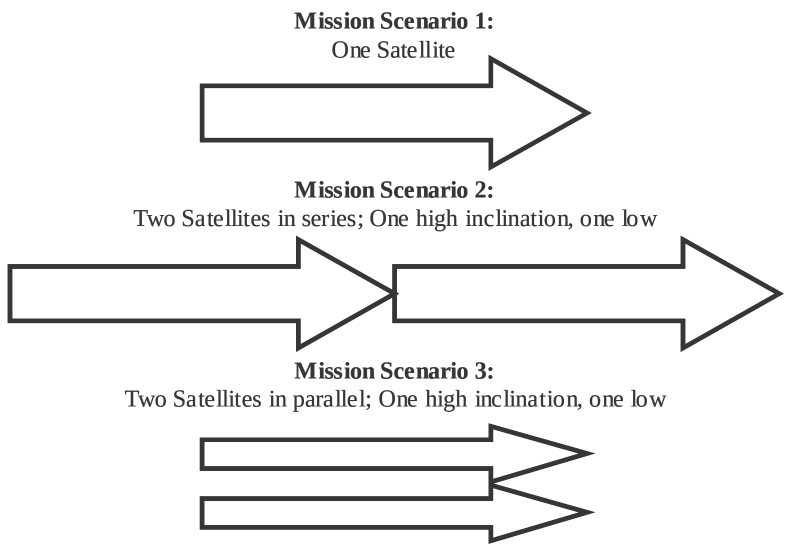

Mission Scenario 1:
One Satellite
Mission Scenario 2:
Two Satellites in series; One high inclination, one low
Mission Scenario 3:
Two Satellites in parallel; One high inclination, one low

Figure 5–1: Mission Scenrio definitions

The first scenario consists of one satellite in either a high or low inclination orbit. In addition to the general rules, the single satellite scenario restricts possible mission to those with a lifetimes greater than six months. Also, the theoretical first unit (TFU) cost of the space craft is less than $100 million.

The second consists of two satellites, one in a high, the other in a low inclination orbit in series (i.e. the second satellite is launched at the conclusion of the predicted lifetime of the first satellite). The additional rules for this scenario require that the each satellite must have a lifetime greater than six months and cost less than $75 million (TFU). The sum of the lifetimes of the two satellites must be greater than 3 years. These rules ensure that the total significant lifetime gain is achieved over the single satellite case.

The third is similar to the second, except that the satellites are in parallel (i.e. the satellites are launched within a month of each other). Unlike the serial case, in addition to each satellite lasting at least six months, the difference between their lifetimes must be less than 2 months. The $75 million dollar cost constrain is retained from the serial case. The new lifetime constraint ensures that a common operation facility can be used for the two spacecraft. If their lifetime were vastly different, the advantage, in terms of operations cost, of having them in orbit simultaneously would be lost.

Once the scenarios are defined, all of the acceptable satellites from the satellite database are matched to each mission scenario in the mission scenario module. In this way, a large database is created that contains the orbital, physical, launch, and scenario information for each mission. Each mission is referenced using an ID number. In this way, computing the cost and utility of an important cross section of the database is feasible.

5.3.3 Evaluation of Missions

Once the mission scenario database is complete, the specifications of approximately 100,000 missions are stored (about 10,000 missions for scenario #1, and about 70,000 missions for scenario #2 and 20,000 missions for scenario #3). The cost and utility modules then independently access the mission scenario and satellite databases and evaluate each of these missions (Note that for scenario #2 a statistical sampling technique was used and only 20,000 missions were evaluated. See chapter 6 for more information). The cost module uses CER's (Cost Estimation Relationships), and NASA's Operations Cost Model to estimate a reasonable cost for each mission. The utility module makes use of orbital and other parameters to determine the usefulness or "utility" of each mission. Both of these modules are discussed in detail in the next section. At the conclusion of the software run, the cost and utility of each of the missions can be plotted and evaluated. Naturally, the most interesting missions are those which have low cost and high utility. The results of the software runs are discussed further in chapter 6.

5.4 Module Descriptions

The modules are listed in the order executed by the software. This order is in accordance with the N^2 diagram shown in Table 5-2.

5.4.1 Orbits Module

Overview
The orbits module creates a database that contains all of the combinations of the orbital design vector (perigee altitude, apogee altitude, and inclination) with a certain resolution. The orbital outputs for each of these combinations are created by calling STK (Satellite Toolkit) reports from within the Matlab code framework. The only derived quantity is the ΔV coefficient. The acceleration due to drag (a dynamic pressure term) is calculated at each point in the orbit. This can then be multiplied by the ballistic coefficient in order to estimate the total ΔV required per orbit to maintain that orbit. Decaying orbits are not considered.

Inputs
The only inputs to the orbits module are the altitude of apogee and altitude of perigee design variables. The radius of the earth and earth's gravitational constant are hard-coded in the module. The altitude of apogee is varied between 200 and 2000 kilometers in 50 kilometer increments. Similarly, the altitude of perigee is varied between 150 kilometers and 350 kilometers with 25 kilometer increments. The inclination is taken at 0°, 20°, 70°, and 90°. Each set of design variables passed to the orbits module is indexed to the orbit database and the corresponding detailed information is passed out.

Outputs

The outputs consist of a variety of STK quantities and one derived quantity, the ΔV coefficient:

Period
Time within the equatorial region (+/- 20^0 latitude) per orbit
Minimum latitude
Maximum latitude
Altitude of apogee
Altitude of perigee
Inclination
ΔV coefficient
Vector of altitudes over a given orbit
Time step (used for orbit propagation)

Assumptions
It is assumed that classical two-body propagation provides orbital outputs that are accurate enough to use in the MATE modeling. It is also assumed that STK provides reliable output for the inputs supplied. The resolution at which the orbital design vector is varied is expected to reasonably capture all the characteristics of each entire parameter space.

Fidelity Assessment
The density model used to calculate the acceleration due to drag over an orbit is quite uncertain. This deficiency is interesting, as it clearly illustrates the lack of high fidelity atmospheric drag models that this mission seeks to remedy. As such, the calculated ΔV can only be assessed as an estimate.

Verification
Later on in the class, a problem was discovered with the altitude vector output from STK. Over the period of the orbit, the altitude vector varied almost sinusoidally by about ten percent of the orbital radius. The orbit was supposed to be a circular orbit, calculated using classical two-body propagation methods without perturbations. Though the cause of this error was not found, the orbital information had to be recomputed. The rather complex Matlab-STK module was later replaced with a Matlab module that used basic two-body propagation to derive the orbital parameters.

The practice of calculating the acceleration due to drag and then using that to calculate ΔV was verified by looking at a 300 km circular orbit. The ΔV required using the standard equation in SMAD and a constant ballistic coefficient matched the value found using this method within about 10 m/s.

An electronic copy of the orbits Matlab module can be found on the XTOS compact disc (labeled Orbits.m)

5.4.2 Spacecraft Module

Overview
The spacecraft module simulates the major subsystems of the actual vehicle. The module contains the following models:

ADACS
CCDM
Payload
Power and Pyrotechnics
Propulsion
Structures
Thermal

In addition to the above subsystems the spacecraft module estimates the ΔV required for orbital insertion and de-orbit. The module calculates the individual subsystem masses as well as the total mass, and estimates spacecraft volume for the launch module.

Inputs
The spacecraft module uses the design vector and the orbital information from the satellite database to complete its calculations. The inputs to the spacecraft module are:

Total Delta-V (200-1000 : 100)
Power (Solar or Fuel Cells)
Communications (TDRSS or AFSCN)
Antenna type (High or Low-gain)
Propulsion Type (Electric or Chemical)
Orbit apogee and perigee.

Outputs
The spacecraft module outputs:
Bus mass
Payload mass
Dry mass
Propellant mass
Total mass
Estimated bus cost
Estimated payload cost
Estimated TFU cost
Estimated software cost
Total volume
Diameter
Length
Lifetime without deorbit
Lifetime with deorbit
Lifetime with insertion and deorbit
Maximum average power required
Maximum peak power required
Latency

Assumptions
The dimensions of the spacecraft are approximated assuming the dimensions of a 2:1:1 cylinder. For lifetime and ΔV calculations, the spacecraft model estimates the ballistic coefficient of drag assuming a Cd of 1.7. This coefficient of drag value is actually lower than the value used in the preliminary ICE design. Three lifetime values are estimated for analysis purposes – one assumes insertion and de-orbit is performed by the spacecraft propulsion system; another assumes that insertion is performed by the IUS and only de-orbit must be performed by the spacecraft; and the final lifetime provides the "ideal" mission life by assuming that insertion and de-orbit both take place without using spacecraft fuel.

Fidelity Assessment
The fidelity on the model is low for a detailed design but adequate for our purposes.

Verification
The spacecraft model was tested and verified using a wrapper script which passed controlled variables into the module for known outputs.

5.4.3 Launch Module

Overview
This module selects the minimum cost launch vehicle for the particular satellite architecture based on a branch and bound algorithm optimized for minimizing cost. The launch vehicle selected is a function of the satellite mass, the stowed dimensions of the satellite, the orbital (perigee) altitude, orbital inclination, and launch site. Each satellite will have a dedicated launch vehicle. Once the launch vehicle has been selected, the total cost of initial deployment is determined based on current cost estimates. In order to minimize computation time in the algorithm, pre-processing was done to determine the injected mass range for the design vector inclinations and perigee altitudes for each launch vehicle. The file containing the post-processed data for selecting the minimum cost launch vehicle is contained in the MATLAB *launch.m* file. The source code for the branch and bound algorithm can be found in the appendix of this document.

Inputs
The launch module post-processed file (*launch.m*), takes inputs from the satellite database. The inputs from the satellite database are as follows:
Inclination (radians)
Perigee Altitude (kilometers)
Satellite Mass (kilograms)
Satellite Diameter (meters)
Satellite Length (meters)

A complete description of the launch vehicles including dimension (fairing diameter and fairing length), as well as cost, is included in a launch vehicle database. The launch vehicles considered are: Pegasus XL (Orbital Sciences Corp.), Minotaur (Orbital Sciences Corp.), Taurus (Orbital Sciences Corp.), Athena II (Lockheed Martin), Delta II (Boeing), and Atlas II (Lockheed

Martin). The launch sites considered are Cape Canaveral (Florida), Vandenberg AFB (California), and Kodiak Island (Alaska).

Outputs
The outputs from the launch module are both final code outputs and are used by other modules. The outputs are as follows:

Launch vehicle
Launch site
Launch cost (nominal)

An error is returned if the input architecture cannot be launched from a U.S. launch site (i.e. too large for a U.S. vehicle or an unreachable inclination).

Constraints
Since this mission is to be conducted by the Department of Defense (DoD), Air Force Research Laboratory (AFRL), only launch vehicles manufactured in the United States as well as launch sites in the United States were considered. Nearly half of the launch vehicles in the database were not considered because of this constraint.

Key Assumptions
Fundamental Equations
This model makes use of the satellite mass, orbital inclination, and perigee altitude to determine the appropriate launch vehicle selection. Satellite mass is the largest driver in launch vehicle selection, sizing, and cost considerations. The pre-processing of the algorithm assumed a rubber spacecraft to determine the maximum injected mass capability. The maximum injected mass capability is 25% greater than the spacecraft mass to account for the deployment cradle. A linear degradation model was available based on an optimal performance inclination for a particular launch vehicle, but was not used in this simulation.

Rationale for simplifications
This model makes use of an approximate satellite mass, orbital altitude, and inclination to determine the launch vehicle selection criteria. Because of the nature of this mission, the number of available launch vehicles was much less than if it were to be a civil launch. Therefore, the small launch vehicles available have a fairly high probability of success; which is why the branch and bound algorithm used the assumption of minimizing only with respect to cost. If the minimization was to be only with respect to risk, the launch vehicles selected may not be available for a particular orbital inclination or would be far more powerful than necessary. It was not practical to use the linear degradation model due to the extremes of our orbital inclinations, 0^0 and 90^0, which are far from the respective performance inclinations.

Evolution of calculations
The pre-processing branch and bound algorithm for minimizing cost, risk, or a weighted combination of cost and risk has remained similar to the B-TOS code. The module used in the simulation (*launch.m*) is much different from the one used in B-TOS. This new module incorporates injected mass capability ranges for each launch vehicle and selects out those

satellites that do not fit in the payload fairing. This module also selects the launch site based on the orbital inclination.

Fidelity Assessment
The costing model contains the same launch vehicle data as B-TOS. The launch vehicle data is the most accurate that could be found and has not changed since the previous version. Because the selection of the launch vehicle is based on minimizing cost, the failure rate of the launch vehicles is not considered. Launch site is considered because of the design inclinations and is incorporated into the module. The branch and bound algorithm does permit launch vehicle selection based on minimizing cost, minimizing risk (or failure rate), or a combination of minimizing cost and risk based on a weighting factor for each minimizing parameter which is determined by the user. If risk were to be a minimizing parameter, it would be computed over a small number of launch vehicles which are very successful or have very few launches which affects the fidelity of the reliability estimate for each launch vehicle.

Verification
The pre-processing of the branch and bound algorithm was done under numerous spacecraft masses, orbital inclinations, and perigee altitudes to determine the range of spacecraft masses that could be launched. The code incorporated the payload fairing dimensions to ensure that the input dimensions of a particular architecture would fit on the selected launch vehicle. Launch vehicles, launch sites, and costs were calculated for various orbital altitudes and inclinations. The post-processed data was cross-referenced with the launch vehicle performance data to ensure that the launch module incorporated the appropriate injected mass ranges for the given orbital altitude and inclination.

5.4.4 Cost Module

Overview
This module calculates the spacecraft costs, the integration, assembly and test costs, the ground support costs, the program level costs, the software development costs, and the operations costs. The module adds these estimations to the launch cost to calculate the total lifecycle costs. Costs are discounted at a 1.9% rate (discount rate typically used by the Department of Defense). They are calculated in FY2000$, and then inflated to FY2002$.

The code uses cost estimating relationships (CER) for small satellites from Space Mission Analysis and Design, 3rd ed., which is solely based on the mass and learning curves, and is valid for spacecraft between 20-400 kg. All of the spacecraft considered for XTOS are within this range. A 95% learning curve on recurring satellite costs is incorporated for missions with more than one spacecraft. The module also uses an estimated value for annual operations cost (before discounting) based on NASA's Space Operations Cost Model (*http://www.jsc.nasa.gov/bu2/SOCM/SOCM.html*). This value is set as a constant in the constants file, and is dependent on the number of satellites operated in each particular year (see constants below for detail).

Inputs

The costing module inputs a vector of up to four satellites from the satellite database. The variables needed for each satellite are as follows:
Apogee altitude
Satellite mass
Launch cost
Lifetime

The module also takes as an input a vector describing the mission scenario. This vector is used to combine the different satellites in the satellite vector correctly when computing the lifetime or operations cost.

The module also uses several constants:

Learning curve slope
Van Allen altitude (for the radiation belt)
Radiation hardening scale factor
Cost for one set of payload instrument suite
Estimated inflation rate (%) (for discounting)
Inflation from 2000 to 2002
Annual operations cost for a one-satellite mission phase
Annual operations cost for a two-satellite mission phase
Number of lines of code (in thousands)

Outputs
The outputs from the cost module are final code outputs. They are:

Theoretical First Unit spacecraft cost (includes bus, software, payload, IAT, PL, and GSE costs)
Cost of all satellites over the whole mission lifetime, discounted (includes bus, software, payload, IAT, PL, and GSE costs)
Operations cost over the whole mission lifetime
Launch cost over the whole mission lifetime
Total non-recurring lifecycle costs
Total recurring lifecycle costs
Total lifecycle cost (spacecraft lifecycle costs, launch costs, and operations costs, discounted)
Total cost error

Fidelity Assessment
The costing module is only as accurate as the models that were used. Its output should be used to compare the relative costs of different architectures, rather than as an absolute number. The error on spacecraft cost typically ranges between 20-40% of the overall spacecraft costs. This error increases with decreasing satellite mass and increasing number of satellites.

Verification
The module was tested over a wide range of satellite lifetimes and masses, for each mission scenario. Realistic outputs and direction of variation of outputs for different inputs were verified.

5.4.5 Utility Module

Introduction
This module is actually a combination of two Matlab modules: the *Calculate_Attributes.m* module, and the *Utility.m* module. Together, these modules collect/calculate the attributes of the satellite architecture in question and compute the single- and multi-attribute utility using data from the MIST interview tool. More specifically, the *Calculate_Attributes.m* function inputs data from the satellite database for the given mission architecture and manipulates that data into an attribute vector. Because attributes can change throughout the mission, one attribute vector is passed for each "phase" of the mission where the attributes remain constant. These vectors are then passed on to the *Utility.m* function, which converts the attribute vector into utility using multi-attribute utility analysis (MAUA). (Please see **the BTOS Architecture Study** for more information regarding MAUA).

Inputs
The *Calculate_Attributes.m* module inputs only two variables:

scenario_cartoon
satellites_used

These variables allow the routine to access the satellite database for each satellite involved in the current mission architecture.

The *Utility.m* module accepts inputs from the following modules:

Calculate_Attributes.m

Very simply, this function inputs data from the utility interview (from *CONSTANTS.m*), and attribute vectors (from *Calculate_Attributes.m*). In addition, the *Utility.m* function calls a small subroutine *Calculate_K.m*. (Note: calling *Calculate_K.m* is only necessary after changes in the interview data occur, and is intermittently omitted from the *Utility.m* module to increase computation speed.)

Outputs
The *Calculate_Attributes.m* module is also very streamlined, and only outputs three variables:

ATTRIBUTE.attribute_values.phase (1 thru 4)
ERROR.scenario
ERROR.mission_life

The first variable is the attribute vector passed to the *Utility.m* module. There is one vector for each phase (for a total of 4), and each one contains all necessary information about the attributes. The two error variables are used to check that the scenario is valid and that the mission phases are calculated properly.

The *Utility.m* module outputs various utility values and error flags:

ERROR.out_of_range – warned if an attribute value was outside the acceptable range
ERROR.single_attribute_util – warned if a single-attribute utility was unacceptable
ERROR.multi_attribute_util – warned if a multi-attribute utility was unacceptable in one of the phases
ERROR.architecture_util – warned if the multi-attribute utility was unacceptable for the entire architecture
UTILITY.single_attribute_util – the single-attribute utility for each attribute
UTILITY.multi_attribute_util – the multi-attribute utility for each phase of the mission architecture
UTILITY.architecture_util – the multi-attribute utility for the mission architecture

Assumptions
There are very few assumptions in these modules, since the modules do not represent actual spacecraft subsystems. However, there are a few assumptions related to the use of MAUA. (Please see **the BTOS Architecture Study** for more information regarding MAUA).
A linear interpolation between points on the single attribute utility curves is assumed.
The use of the functional form of utility is based on two key assumptions:
Preferential Independence: The preference of $(X_1', X_2') > (X_1'', X_2'')$ is independent of the level of $X_3, X_4, \ldots, X_n$.
Utility Independence: The "shape" of the utility function of a single attribute is the same, independent of the level of other attributes. "Shape" means that the utility is the same up to a positive linear transformation, $U'(X_i) = aU(X_i) \pm b$.

Fidelity Assessment
There is some uncertainty in the utility values due to the method and tools chosen for the utility interview. The MIST measurement resolution for a single point on a single-attribute utility curve is ±0.05. The MIST measurement resolution for the weighting factors (k values) is ±0.025. These values are solely due to the resolution of the MIST software tool. If the customer is "uncertain" about his preferences, these utilities will shift as well (assumed to shift by the same amount as the software uncertainties). Sensitivity analysis reveals that a worst-case decrease of 0.1 (twice the software uncertainty) in all single attribute utilities will produce a drop of 0.1 in multi-attribute utility. Likewise, a drop of 0.05 (twice the software uncertainty) in all k values results in a drop of 0.075 in multi-attribute utility.

Verification
This code was verified with many test cases and imaginary mission architectures. The code is capable of handling mission scenarios involving multiple satellites in parallel or in series, but these mission scenarios were never run due to time constraints.

5.5 Integration Process

The integration of the modules mentioned above into one piece of stand-alone software was a long and arduous task. A brief description of the coding conventions that were used is described below, as well as the N^2 diagram that determined the proper execution order of each module in the sequence. Despite the efforts of the integration team to streamline the integration process, a few hang-ups were encountered. The lessons learned from those complications are also captured below.

5.5.1 Variable and Module Conventions

The use of global structures to store the spacecraft architectures and relevant outputs from each module greatly reduced the need for strict variable naming conventions and input/output standards. The satellite database modules (orbits, spacecraft, and launch) required very few inputs to complete their calculations, and the modular nature of the code allowed each module author to name the inputs by whatever name he/she wished within the code. The integration team simply had to ensure that the correct elements of the satellite database were passed to each module in the order it requested them. Passing conventions were even less important in the cost and utility modules, as they were simply given access to the entire satellite database. Authors simply relied on a list of database element names to pull the desired parameters. All modules were designed to pass their outputs in the form of structures, with each output structure titled by the name of the module where it originated (i.e. spacecraft module output the SPACECRAFT structure). Again, the use of global structures allowed the integration team to call the outputs of each module by whatever name they desired when storing the data to the database. A sample of input/output conventions follows:

Sample function call – spacecraft passes outputs to the structure SPACECRAFT.

```
[SPACECRAFT] = spacecraft(SATDB.power_type, etc…);
```

Sample database entry – store mission_life from SPACECRAFT to the element lifetime in the SATDB (satellite database) structure.

```
SATDB.lifetime = SPACECRAFT.mission_life;
```

This method drastically reduced the time spent on verifying variable names and checking homogeneity. Users who accessed the satellite database later on simply needed a list of the parameter names to pull the desired information for a particular satellite.

5.5.2 N^2 Diagram

To prevent implicit loops that could arise by calling a particular module before calling the module it was dependent on, an N^2 diagram needed to be constructed. Figure 5.5.1 represents the N^2 diagram used by the integration team to assemble the main execution sequence. The diagram can be read as follows:
Reading a particular row from left to right reveals the modules that the module in that row depends on for inputs. For example, reading the “Spacecraft” row from left to right reveals that the module relies on inputs from the design vector (Design), the constants module (Constants), and the orbits module (Orbit).
Reading down a particular column reveals the modules that the module in that column passes outputs to. For example, reading the “Launch” column down shows that the costs module (Cost/Schedule), the utility module (Utility), and the outputs of the software (Outputs) have a direct dependence on the outputs of the launch module.

Table 5-2: XTOS MATE N^2 Diagram

XTOS MATE Modules N^2 Diagram	Design	Constants	Orbit	Spacecraft	Launch	Cost/Schedule	Utility	Outputs
Design	■							
Constants		■						
Orbit	I	I	■					
Spacecraft	I	I	I/O	■				
Launch	I	I	I/O	I/O	■			
Cost/Schedule	I	I	I/O	I/O	I/O	■		
Utility	I	I	I/O	I/O	I/O		■	
Outputs	I	I	I/O	I/O	I/O	I/O	I/O	■

5.5.3 Lessons Learned

Although the MATE modules were improved dramatically over previous versions, the code framework and the integration process still left something to be desired. In particular, the time constraints of the class did not allow for higher fidelity modules to be developed. Consequently, some architecture options must be left out of the tradespace exploration because they exceed the achievable complexity of the modules. In the future, a greater range of possibilities might be modeled if the span of the class is extended or if the coding process could begin earlier in the class.

Another important lesson noted during the integration process was the desperate need for some sort of Matlab/computer modeling familiarization at the beginning of class. A great deal of confusion arose out of a general lack of coding experience among the class members. As stated earlier, the presence of previous versions of the modules provided an excellent foundation for building the new modules, but many modules still required a great deal of modification. Future architecture studies should consider the use of one or two class sessions to familiarize the students with the coding environment as well as with a basic approach to computer modeling.

Finally, the exact framework and *software* architecture must be decided on far in advance of the actual start of the coding process. Unfortunately, the database concept was not decided on until many teams had already begun the coding of their modules. That late decision resulted in extra work for the integration team that could have been avoided if the framework principles had been communicated at an earlier time. Nonetheless, the integration and software computational time saw a dramatic improvement over previous versions; the consideration of these lessons should MATE-CON

6 MATE Code Results and Sensitivity Analysis

6.1 Tradespace Enumeration

Since the code was highly modular and much of the work had been done in creating the SatDB, the single satellite case took about 15 minutes to run. The multi satellite cases took longer but were each completed within approximately two hours.

6.1.1 Entire Tradespace

After running the code, the following tradespace graphs were produced:

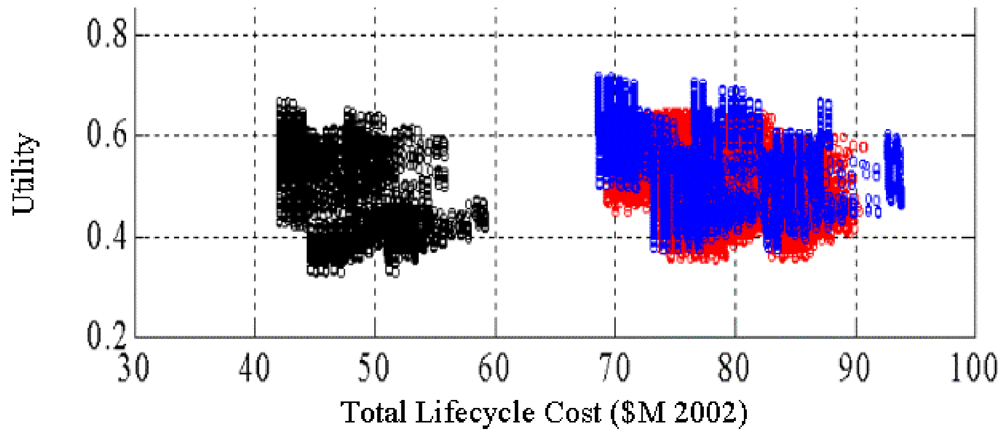


Figure 6–1: Complete tradespace

In Figure 6–1, we see three distinct groupings of architectures. The black architectures represent a single satellite, the red represent two satellites launched in sequence, and the blue represent two satellites launched in parallel.[7]

Due to a lack of time and the amount of increased complexity with multiple satellites, we decided to analyze only the single satellite case. Analyzing the more complex multiple satellite cases added no apparent value to the learning experience, whereas exploring the single satellite case in detail certainly did. Furthermore, the user did not express interest in the additional utility gained by the multi-satellite mission given the nearly doubled cost.

[7] As mentioned earlier statistical sampling was used to reduce the computational load for the “two in series” scenario. Progressively larger and larger samples were taken until the pareto optimal front did not appear to change. Furthermore, the minimum scenario length of 3 years eliminated cases of two satellites in orbit for a short time in low orbits (e.g. a year total). Such a situation would have yielded high utility.

6.1.2 Single Satellite Tradespace

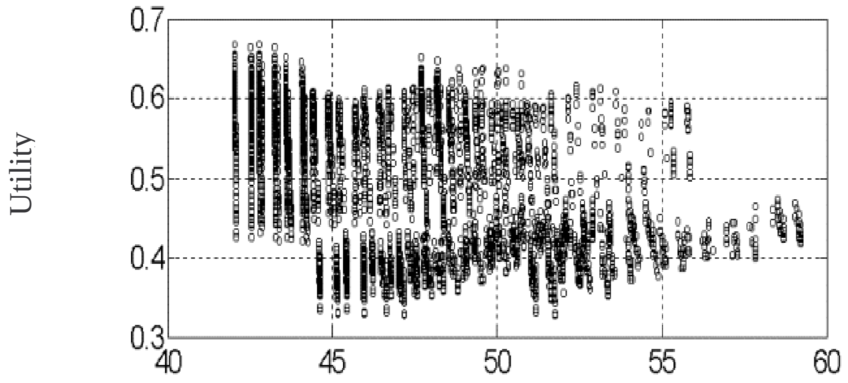


Figure 6–2: Single Satellite architectures

Figure 6–2 shows the single satellite case's tradespace. Several distinct bands can be seen. One of these bands begins at approximately ($51M, 3.3) in the space and continues up and to the right on the graph. This pattern of increasing utility with increased cost is repeated throughout the plot until the top.

At the top, we get a different case. There is a beginning of the regular pattern, but it is truncated before it can rise in cost and utility. The class believes that the tradespace was restricted at that point due to two factors: the customer's requirement that this mission be launched on a small to medium U.S. launch vehicle and the class's estimate that, to remain within the constraints of these launch vehicles, the satellite would not be able to carry more than 1,000 meters of Delta V. These higher cost, higher utility architectures would come about if we were able to lift more fuel into orbit, thus simultaneously enabling a longer lifetime and a lower orbit.

Due at least to this constraint (and possibly to other factors) we have an unusual pareto frontier. Normally, one would expect to see a line of architectures that generally covers the left hand side of the plot where one could receive a higher utility for a higher cost. In other words, one would expect an architectural tradespace to show that trades could be done to increase the utility. This is the first logical thought that leaps to mind when considering expense: if one desires a better product, one usually has to pay more for it. However, in this particular case, there is a clear winner: there is a single lowest-cost, highest-utility point on the graph.

6.1.3 Apogee and Perigee

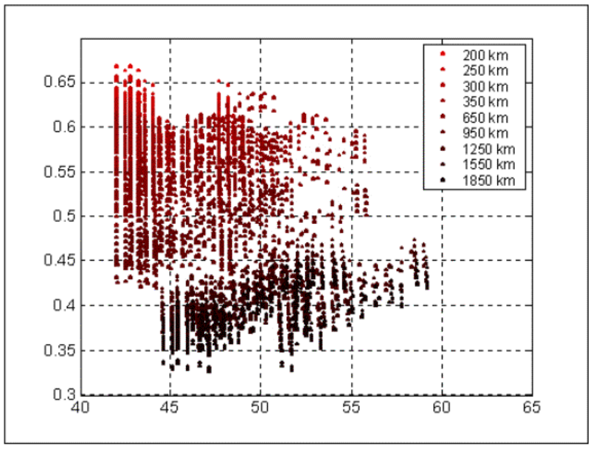


Figure 6–3: Variation due to apogee altitude

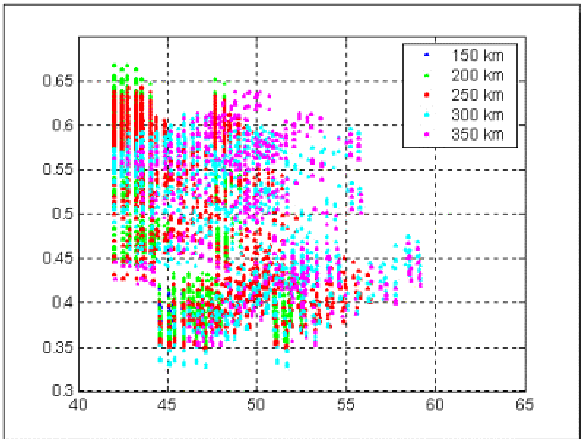


Figure 6–4: Variation due to perigee altitude

Figures three and four, when viewed together, tell a story. Figure three is the single satellite utility plot with each architecture colored by its apogee altitude. A strong inverse correlation can be seen, with lower apogee altitudes having a higher utility and vice versa.

Figure four is a very similar plot, but with the coloring done by perigee altitudes instead of apogee. There is no strong correlation shown, but rather a seemingly chaotic mishmash of colors. Further study was needed to understand this correlation.

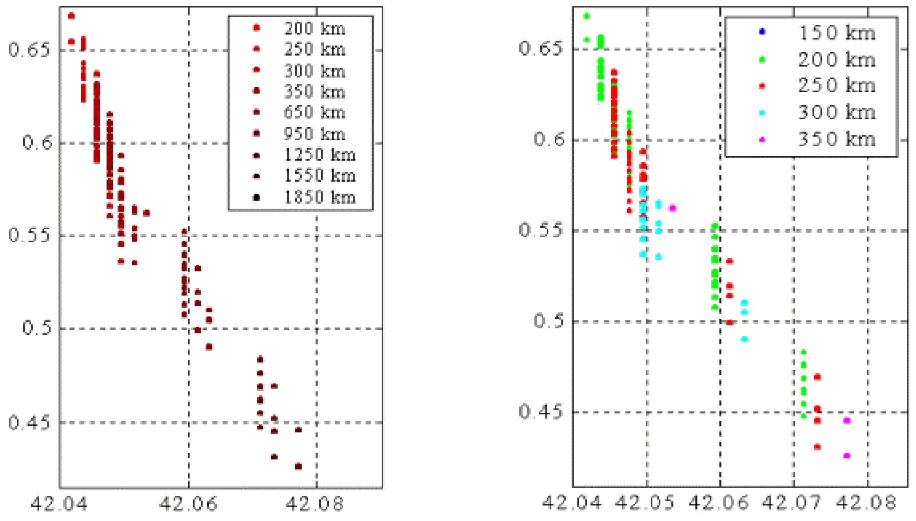


Figure 6–5: Zoom in on left edge

Figure 6–5 shows two enlarged plots of the same architectures (zooming in on the narrow column of points at the far left), with the left plot colored by apogee and the right by perigee. As can be seen, the architectures fall into small micro blocks. As can be seen by the apogee chart, it is the apogee that determines in which block an architecture falls.

The same plot, this time colored by perigee, shows that perigee determines the order of the architectures *within* these blocks. Within each micro block is seen the same sort of pattern that we saw within the macro diagram: increasing utility with decreasing perigee. Thus, it is clear that perigee is a secondary driver of utility while apogee is the primary driver of utility.

6.1.4 Lifetime

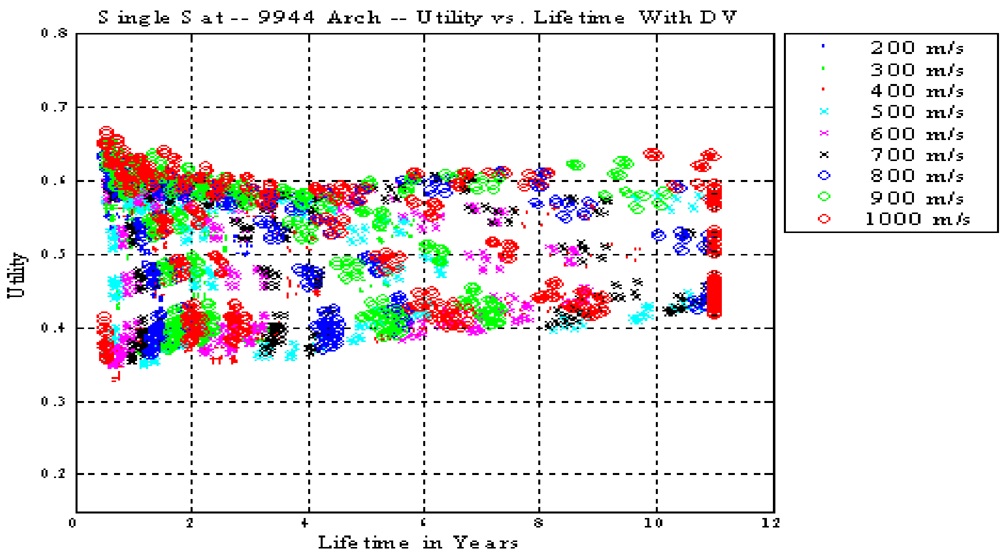


Figure 6–6: Utility vs. lifetime; colored by Delta-V

Finally, a look at lifetime is needed. Figure 6–6, above, a plot of lifetime vs. utility with the points colored by the amount of Delta V carried, shows the reader another apparent mishmash of color. This plot, however, rewards careful consideration. There are bands of color that start nearly vertical on the left and rapidly decrease to a negative slope as the bottom of the band goes further out to the right.

These bands of color represent isometric lines of delta-V. Within one of these bands, one can increase utility and decrease lifetime by reducing the apogee and perigee of the orbits, which moves the architecture up and to the left along the isometric line. It is clear from this that lifetime is a secondary or possibly even a tertiary driver.

This is to be expected, as it comes directly out of the value given by the customer on 'Data Lifetime,' one of the user defined attributes. In the original utility function, which was later changed (see Chapter 9.1), Data Lifetime was rated as the least important of the attributes, so it is logical that lifetime would not be a primary driver.

6.2 Sensitivity Analysis

The purpose of a sensitivity analysis is to determine how sensitive a model is to changes in its inputs. Due to time constraints the class was unable to perform variations in all of the variables (nor should such a study be necessary, so time was spent determining which variables were most uncertain, and a sensitivity analysis was done only on the most uncertain variables.

6.2.1 Determination of uncertainties

There are two kinds of uncertainties in X-TOS: model uncertainty and preference uncertainty. Model uncertainty refers to the uncertainty due to estimations and errors in the model. Preference uncertainty refers to the uncertainty of the tradespace. Figure 7 shows a way of thinking about this:

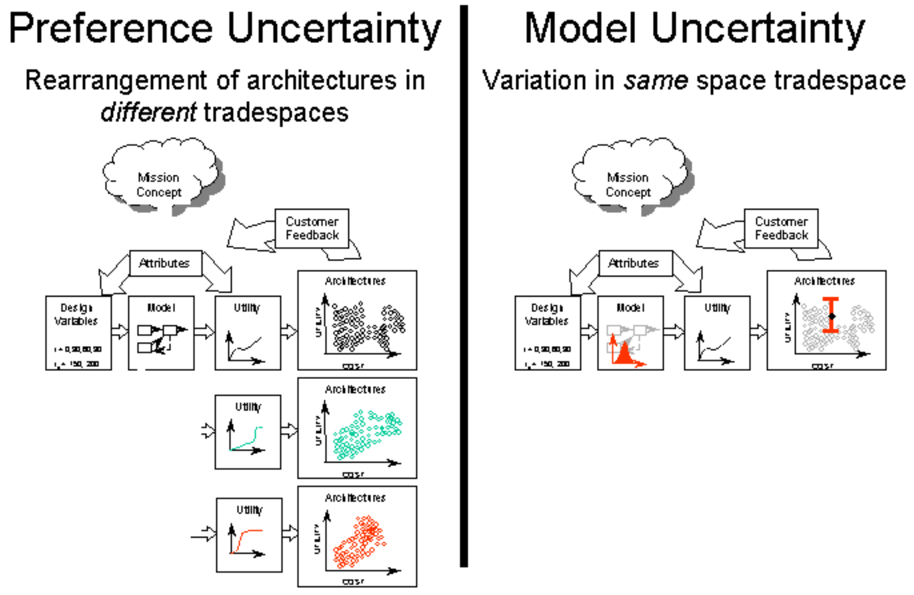


Figure 6–7: Preference vs. Model uncertainty

As can be seen, preference uncertainty can be thought of as something that changes the entire tradespace, whereas model uncertainty can be thought of as something that affects the architectures within a tradespace.

6.2.2 Model Uncertainties

Due to time constraints, the class decided to look only at model uncertainty. A particular case of preferences changing is discussed in Section 9.1. As the models used were parametric, a close look at the effects of parametric modeling on uncertainty was in order:

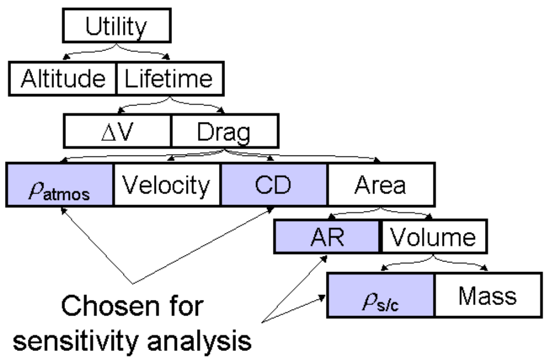


Figure 6–8: Uncertainty decompostion for utility

Utility (referring to the revised utility function see Section 9.1) is the parameter chosen to examine. It is primarily driven by altitude and lifetime. Altitude is known with reasonable certainty, where as lifetime is quite uncertain. Lifetime is primarily composed of delta V and the force slowing the spacecraft down (mostly drag). The equation for drag on a body is well known and is composed of the density of the ambient fluid (the atmosphere), the velocity of the body, the coefficient of drag, and, within hypersonic particulate flows such as this one, the cross sectional area. Within the class's models, the density of the atmosphere and the coefficient of drag provide large sources of uncertainty, as does the cross sectional area. However, the area can be broken down further into the aspect ratio (a number that was chosen to facilitate launch vehicle choice while still retaining enough detail to calculate drag) and the volume of the craft. Volume was calculated by using standard spacecraft densities and the calculated mass. Since dry mass and fuel are accounted for separately in computing volume, changing the spacecraft density only affects the volume occupied by the 'dry' portion of the spacecraft.

6.2.3 Sensitivity Analysis

These four chosen parameters were chosen for the sensitivity analysis as they were the most uncertain: the density of the atmosphere, the coefficient of drag, the aspect ratio, and the density of the spacecraft. These four were varied, one at a time, to find the effects on the representative architectures:

6.2.3.1 Spacecraft Density

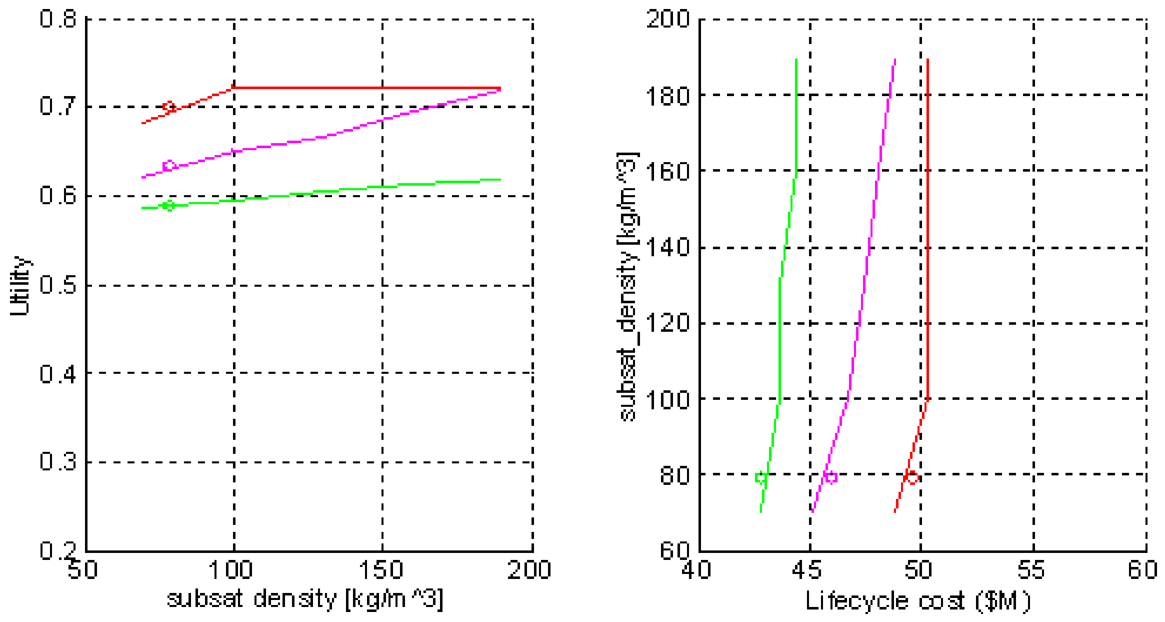


Figure 6–9: Sensitvity to Spacecraft Density

Figure 6–9 shows the effect of varying satellite density on three representative architectures. The range over which spacecraft density is varied was developed by taking the satellites designed in the ICE process and back calculating their densities. This calculation gave densities in the 150 to 200 kg/m^3 range. The circles represent the base values used to evaluate the remainder of the tradespace. As can be seen, the green architecture varies only slightly if density is increased, the magenta architecture varies significantly, and the red architecture initially varies even more strongly than the magenta, but quickly reaches a maximum.

The reasoning for this is clear when one understands what these three representative orbits represent. The green architecture has a low orbit relative to red and magenta, and gains most of its utility from its altitude. The magenta orbit has a middle orbit and gains some utility from its orbit and some of its utility from lifetime. The red architecture has a high orbit and gains most of its utility from lifetime.

Knowing this, the graphs become quite intelligible. Increasing a spacecraft’s density increases its efficiency in traveling through the atmosphere and gives it a longer lifetime. Since the green architecture derives little of its utility through lifetime, increasing its lifetime increases its utility only a small amount. The magenta architecture can gain quite a bit of utility by increasing its lifetime. The red architecture, though, reaches its maximum utility lifetime of 11 years quite rapidly and can gain no more utility from increasing its lifetime. (See section 3 for an explanation of the maximum utility from lifetime.)

Cost is seen to increase slightly as mission life increases. This is due to the increased operational costs associated with a longer lifetime.

6.2.3.2 Aspect Ratio

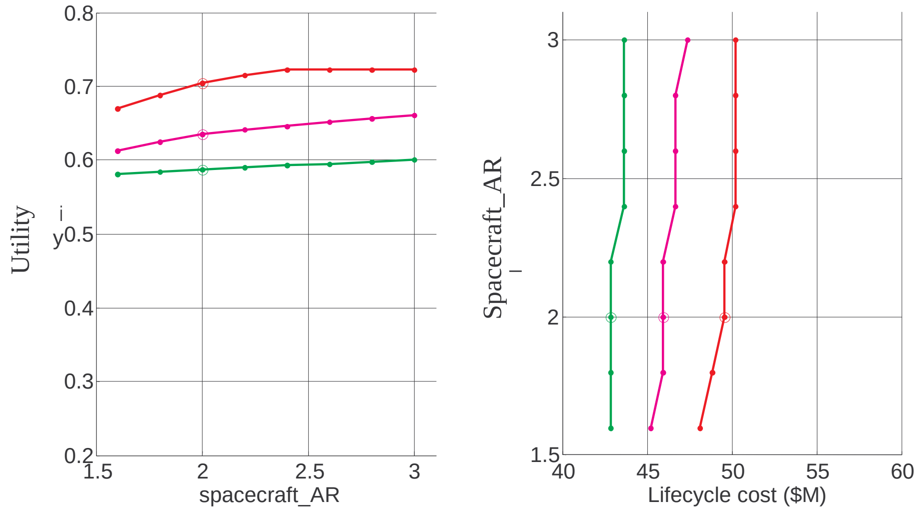


Figure 6–10: Sensitivity to Aspect Ratio

Similar things are seen in the aspect ratio graphs: Green’s utility increases the least, magenta’s is in the middle, and red’s increases most rapidly until the 11 year lifetime is reached. Similar cost increases are seen due to the increased lifetime.

6.2.3.3 Coefficient of Drag

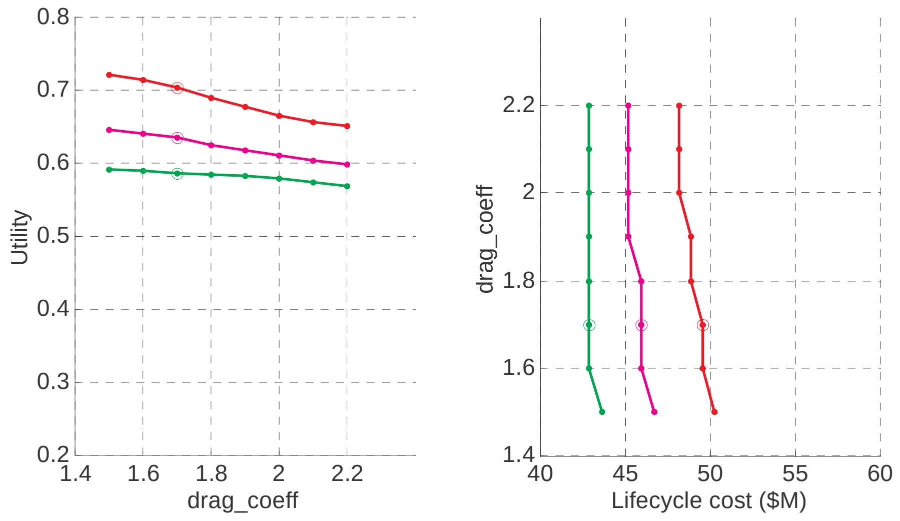


Figure 6–11: Sensitivity to drag coefficient

In the case of coefficient of drag, we see that a decreasing CD increases utility and cost. We see that the three architectures respond in roughly the same ways as in the previous two sensitivity analyses.

6.2.3.4 Atmospheric Density

The case of atmospheric density is not so simple. There are two different causes of uncertainty here: it is uncertain at which point in the solar cycle the satellite will be launched, and the current models of atmospheric density have model uncertainty associated with them. First, the uncertainty due to the solar cycle will be examined:

Solar Cycle uncertainty

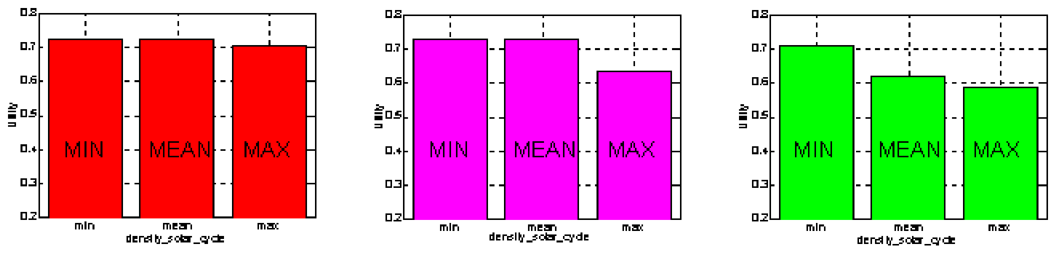


Figure 6–12: Sensitivity to Solar cycle variation

The colors on the graphs in Figure 6–12 represent the same red, magenta, and green architectures. The three bars on each graph represent the utility of the architecture at solar max, solar mean, and solar min. (It should be noted that, for simplicity, the model assumes that these are the cycle conditions that the satellite will face all day of every day of its orbit.)

It can be seen that if the red architecture were launched into solar max, it would have a slightly smaller utility than if it were launched into solar mean or solar min. This is because the changes in the density of the atmosphere at higher altitude is very small on an absolute scale. Magenta's utility decreases more at solar max due to the atmosphere's larger absolute increase in density at that altitude. Finally, the green architecture decreases rapidly as the density increases, as the solar cycle's absolute effect on atmospheric density is orders of magnitude larger than at the higher altitudes.

Atmospheric density model uncertainty:

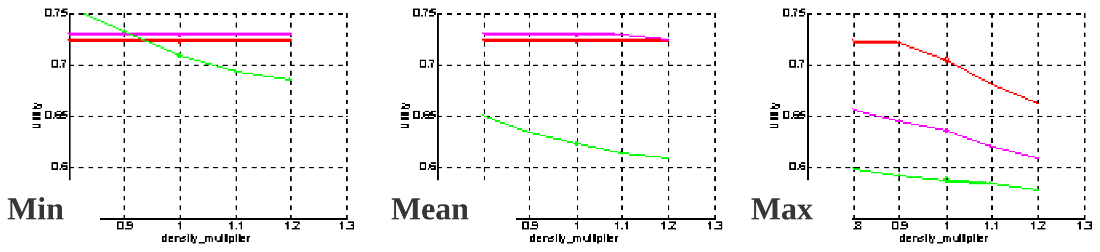


Figure 6–13: Sensitivity to atmospheric density (model uncertainty)

When examining the effect of drag model uncertainty on utility, one can see that this mission's results—improved drag models—are needed to determine which architecture to choose. If the drag at solar min is small, then the green architecture's utility surpasses both the red and magenta's; in the case of solar mean and solar max, the green architecture is the clear loser.

6.2.4 Insights and Conclusions from sensitivity study

What arises from considering this sensitivity analysis is fairly remarkable. In planning to design the satellite so that it is robust towards these uncertainties, the class discovered that by adding more fuel one has the freedom to *dynamically* change the orbit based upon the data collected.

That is to say, once the satellite is in orbit, the massive amount of fuel that has been loaded to make the design robust to uncertainty can be used to change its orbit to be appropriate to the solar cycle and maximize utility. If, as the data comes in, it becomes apparent that more utility could be gained by having the satellite at a different orbit, the satellite can be moved into another orbit.

This clearly shows us the importance of considering model uncertainties in architectural tradespace exploration. Not only can the design become more robust to uncertainty, but new insights into the system can be gained. These insights can lead to more creative solutions to the problem at hand, thus more effectively utilizing the resources available to the stakeholders involved.

7 MATE-CON

7.1 What is MATE-CON?

Multi-Attribute Tradespace Analysis with CONcurrent (MATE-CON) engineering is a process allowing subsystems to trade design parameters in a formal setting. It provides real-time feedback into the effect of those trades on other subsystems and the overall utility of the mission. It facilitates detailed analysis of the tradespace by automating much of the trade environment while maintaining the human interaction that is the power behind concurrent engineering. It is faster than traditional processes because the system can be set up such that all design values are calculated based on given input. This allows for quick examination of the results of varying certain input parameters. This is primarily a design tool for early design phases. Once the system has gone into the detailed design phase the number of possible trades drops drastically.

7.2 Human Interaction

Part of the power of a system like this one is that all engineers operate within the same environment both physically and technically. They share the same software tools and can communicate easily within design sessions. Human interaction is key to this process. It allows for checking of computer results and easy verification that other engineers are producing what you expect. Design sessions are "scripted" and controlled by one person allowing for a more organized setting. This process forces many eyes on the final product both in the overall design and in the individual subsystems. The reason for this is that many other subsystems are affected by changes in another. For example, during an XTOS session, errors in the ADACS subsystem were discovered by excessive power requirements given to the Power and Pyrotechnics subsystem. Experts in each area play with key trades that directly affect their subsystem and the overall spacecraft. Those directly responsible for particular areas know best which trades have the greatest effect on their subsystem. Each subsystem produces trade trees showing these effects.

7.3 Utility Feedback

Another testimonial to the power of this method is the fact that the engineers can utilize the results of the MATE process to provide a real-time view into the utility of their new system. The software developed in the first half can be integrated into the new Concurrent engineering environment to analyze the design within the utility vs. cost space. Each design is placed back into the tradespace from MATE to see if it has moved with respect to utility and these parameters can be analyzed to determine whether or not this design is an improvement.

7.4 Spacecraft Subsystems

7.4.1 Introduction

Multi-Attribute Tradespace Exploration with Concurrent Engineering (MATE-CON) utilizes the working knowledge of spacecraft subsystem specialists. Through the use of a software tool that interacts with Microsoft Excel, called ICEMaker, the MATE-CON process is translated into a preliminary design tool. Each spacecraft subsystem specialist is responsible for an Excel workbook that interfaces with the other subsystem workbooks through the ICEMaker software. Each workbook has an Outputs worksheet and an Inputs worksheet. The subsystems are responsible for publishing their respective Outputs to the ICEMaker server. Publishing the Outputs to the server makes the variables available to all the subsystems, and in turn the subsystems request the published variables through their Inputs worksheet. Once an output on a single sheet is changed, it is an iterative process of publishing and requesting of all the subsystems to converge on a single design. A detailed synopsis of each subsystem follows.

7.4.2 Systems

Introduction

The Systems subsystem can be described as the “control” subsystem. Within this workbook is contained a mass summary and breakdown; a power summary and breakdown; and a sheet capturing the main items of the other subsystem sheets to verify that data is being passed correctly. These sheets are checked at each iteration of the ICE process, and the total system mass, both dry and wet, with contingency and without. Another important set of outputs are the contingency levels for each section of the satellite. The inputs far outnumber the outputs in this subsystem, as the systems chair monitors the progress of the design. The various charts (power, cost, and mass breakdowns) are projected on the video screens and verified and compared with the previous iteration. The Systems chair is responsible for calling convergence of the design.

Inputs

The input list for the Systems subsystem is fairly long; more than ¼ of the total inputs are taken by the chair. Inputs are drawn from every other subsystem and used for verification and calculation of the power, cost, and mass breakdowns.

Outputs

Compared to the inputs list, the outputs list is small. Yet these outputs are quite important, including the current total mass for the system (with and without contingency), average and max power per mode after adding contingencies, cost, and reliability numbers.

Assumptions

There are few assumptions made for the Systems module. Mainly, the contingency numbers are based on SMAD’s contingencies for a preliminary design, but as this preliminary design is fairly detailed we have reduced the mass and power budgets to 15%. The design is fairly robust to mass margin reduction—with so much fuel, the final vehicle might come in over weight. That simply means that less fuel would be loaded thereby reducing lifetime but allowing launch.

Fidelity Assessment

As the Systems chair takes so many variables from other systems and performs a relatively small number of simple calculations, the fidelity of the chair is dependant only upon the fidelity of the other chairs’ calculations.

Verification

Values in this subsystem were verified by hand calculations. Heavy communication between Systems and the other subsystems helped to point out any inconsistencies when passing variables. Errors were therefore easy to find and fix.

7.4.3 MATE-CON Chair

Introduction

The MATE-CON Chair is a new addition to the ICE process. It can be described as the link between MATE and CON. Contained within this workbook is an Excel to Matlab link that allows the running of the utility code (developed under MATE) with inputs taken from the current design in the ICE Session. The purpose of this chair is to assist the systems engineer in directing the trades in the design—as changes are made, the utility can be tracked and the proper direction for the design can be determined.

Inputs

The input list for the MATE-CON chair is short, and includes only those parameters from the ICE session that are needed for utility calculations, i.e., the parameters that are used to compute the attributes.

Outputs

The outputs list is also fairly small for this chair and includes the utility for each attribute as well as the overall multi-attribute utility.

Assumptions

The assumptions made here are the same assumptions made in the earlier MATE section of this paper.

7.4.4 Mission

Introduction

The mission subsystem can be described as the "primary" subsystem. Within this workbook is a shortened list of the design vector (from the MATE Matlab modules). These design variables are changed by hand for each iteration of the ICE process, and the outputs are then sent to every other subsystem. In addition to the design vector, the mission subsystem also calculates an assortment of mission parameters, such as orbit characteristics, launch vehicle characteristics, delta V budget, and spacecraft lifetime. The outputs far outnumber the inputs in this subsystem, as the spacecraft lifetime and delta V budget are the only two calculations requiring inputs from other subsystems.

Inputs

The input list for the Mission subsystem is significantly shorter than most. Inputs are drawn from the System, MATE-CON, Configuration, and Propulsion subsystems, yet all of these inputs are used for just the lifetime and delta V budget calculations:

- *Delta V* inputs from propulsion (for Stationkeeping, ADACS, and contingency) are combined with internally calculated values for insertion and deorbit to produce a delta V budget.
- The lifetime calculation incorporates many variables such as *coefficient of drag* and *cross sectional area* from Configuration, total wet mass from Systems, *Stationkeeping delta V per orbit, per BCD* from MATECON, and *total propellant mass, propellant mass per orbit,* and *specific impulse* from Propulsion.

Outputs

Compared to the inputs list, the outputs list is very extensive. Yet these outputs can be partitioned via the calculations that created them. The output types of each are:

- Design Variables - these outputs require no calculations and were changed many times during the iteration process: *perigee altitude, apogee altitude, inclination*, and *total delta V*.
- Launch Vehicle Selection – these outputs are directly related to the choice of launch vehicle: *launch vehicle type, payload capacity, payload dimensions, launch environment, cost, reliability, insertion error*, and *mass*.
- Orbit Determination – these outputs are directly related to the chosen orbit: *orbit parameters, eclipse time,* and *orbit period.*
- Calculations – these are simply the *lifetime* and *delta V budget* outputs calculated in the workbook.

Assumptions

When trying to find data on our primary launch vehicle, the Minotaur, we ran into a few obstacles. Because the Minotaur uses an ICBM as a lower stage, we could not find an accessible payload planner's guide or something similar. And with its first launch in the year 2000, there is very little historic data to pull from. Consequently, some of the values for the launch environment were assumed from current models of similar launch vehicles (namely the Taurus and Pegasus). These values are marked in the worksheets. In addition, J2 effects are not included in the lifetime calculation, due to the fact that a precessing orbit has negligible impact on our specific mission.

Fidelity Assessment

The largest possible source of error in the mission subsystem is in the launch environment assumptions. Yet these values are passed only to the structures subsystem, where they are used in a precautionary analysis to ensure our spacecraft survived the launch phase. Changes in these values have a minimal effect on the spacecraft as a whole. There is also uncertainty in the lifetime calculation, which is found by burning the available fuel for stationkeeping and ADACS, until there is none left. However, many of the inputs for this calculation are conservative (such as Stationkeeping delta V per orbit/BCd, specific impulse, and total usable propellant mass) and thus the calculation for lifetime is conservative as well.

Verification

Values in this subsystem were verified by hand calculations. In addition, heavy communication between Mission and directly related subsystems helped to solve any inconsistencies when passing variables.

7.4.5 Payload

Introduction

The payload sheet functioned to directly translate the specifications of the three instruments from their respective requirements documents to the ICE environment. The three instrument components of the payload were: 1) Satellite Electrostatic Triaxial Accelerometer (SETA), 2) Absolute Density Mass Spectrometer (ADMS), and 3) Composition and Density Sensor (CADS).

Inputs

The payload subsystem uses the following inputs:

- Power mode definitions, which were used to calculate the instrument power requirements for each phase of flight

Outputs

The payload subsystem output the following:

- Mass, dimension, and location and requirement for each instrument
- Combined pointing requirement
- Peak and average power requirements for each power mode
- Number of redundant instruments
- Failure rate of each instrument

Assumptions

The only assumption made was that the requirements document was accurate in its portrayal of size, shape, and other specifications of the instruments.

Fidelity/Verification

This is a function of the requirement document's accuracy.

7.4.6 Configuration

Introduction

The configuration subsystem arranges each of the subsystem components on the spacecraft. A very useful tool for this arrangement is DrawCraft.8 The subsystem chairs publish the dimensions, mass, and locations (if applicable) of each of the components using ICEMaker. Next, these values are automatically updated to a SCMS (Shared Mechanical Control Sheet) text delimited file, which is then read by DrawCraft. DrawCraft then creates an assembly in SolidWorks which provides information on the weight distribution and surface area over the entire spacecraft.

The components need to be placed in such a way that certain criteria are met. Since this spacecraft is traveling through a significantly dense part of the atmosphere, it needs to be aerodynamically stable. In this case, the center of gravity needs to be forward of the half-chord point. For our purposes, the length of the spacecraft is approximated as its chord. Also, since the scientific sensor suite was previously chosen, the requirements of the sensors need to be met. The ADMS and CADS sensors are required to be ram-facing, and the SETA sensor is required to be within six inches of the center of gravity of the entire spacecraft. Another important requirement is that the entire vehicle needs to be able to fit inside the payload fairing for the chosen launch vehicle.

This subsystem is built so that with a small amount of human involvement, the configuration of the satellite can be dynamically changed during the ICE sessions. Human involvement is required for several different reasons. First, the updated SCMS needs to be loaded into DrawCraft. Once DrawCraft creates the model in SolidWorks, one needs to open a special window within SolidWorks to produce the weight distribution and surface area outputs. DrawCraft does provide some of these required outputs; however, SolidWorks provides all of the required outputs, and does so in a favorable manner. For example, the moments of inertia calculated by DrawCraft are about a reference axis, and the moments created by SolidWorks are both around a reference axis and the center of gravity. Since the center of gravity changes with every design iteration, SolidWorks is a more useful tool. It is conceivable that this type of program interface could be automated so that the SCMS file is automatically updated, and the information is automatically published from SolidWorks. This would aid greatly in the speediness of the ICE sessions. It was not developed in this case because of time constraints.

Even if the SCMS file could be automatically updated, and the outputs automatically published, it would only take care of parametric variations on the design. The configuration subsystem is unique in that at each iteration step in the ICE session, the configuration needs to be visually evaluated and possibly changed by the configuration chair. A good example of this is that at the start of the ICE sessions, the original design for the fuel tank was a single sphere. As the fuel mass increased, the fuel tank impinged upon, then eventually exceeded, the wall of the main bus. The result was that a non-parametric change to two cylindrical tanks needed to be made, as can

[8] DrawCraft - Dr. Joel C. Sercel (Caltech, Pasadena, California, USA) in the Laboratory for Spacecraft and Mission Design for the use of the DrawCraft (a spacecraft configuration tool).

be seen below in figure 1. Another example that illustrates the necessity of configuration evaluation at each step concerns the scientific sensors. Once the change to cylindrical fuel tanks was made, a trade was performed in which the satellite altitude was lowered. This required more fuel to be aboard, and the fuel tanks to lengthen. Eventually the tanks, though they fit inside the main bus, encroached upon the space needed for the scientific instruments. This can only be seen when the configuration chair takes the time to visually evaluate the design. In this trade, the constraining factor happened to be the space required for the fuel tanks. If the configuration were not evaluated visually, an impossible design could be chosen.

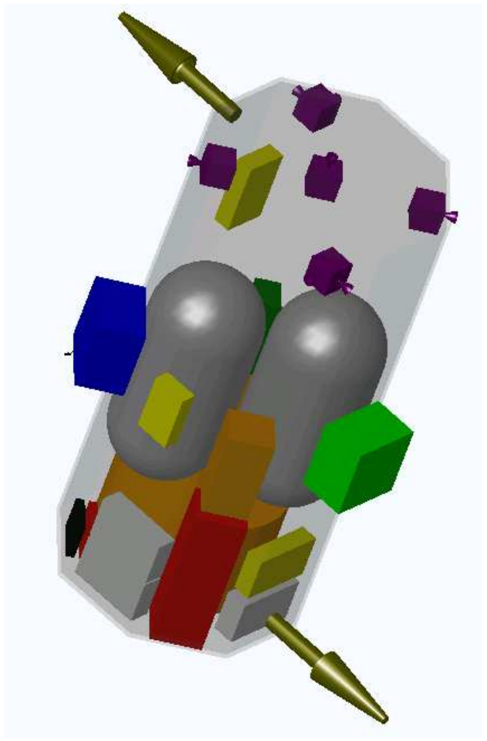


Figure 7–1: Final design. Note cylindrical fuel tanks (grey)

Inputs

Generally, the inputs to the configuration subsystem are the dimensions, mass, and location of each of the satellite components. The components that were modeled were:

- Main bus
- CADS, ADMS, and SETA sensors,
- Omni-directional antennas
- Primary and secondary batteries
- Fuel tanks
- ADACS thrusters
- Main Thruster
- Telecom boxes
- C&DH computers

Outputs

The parametric outputs of this subsystem described the weight, surface area, and volume of the total spacecraft. These are listed below:

- Basic cross-sectional shape of the main bus
- Basic shape of entire bus
- Cross sectional area
- Total surface area
- Coefficient of drag
- Distance from the c.g. to center of aerodynamic pressure
- Distance from the c.g. to center of solar radiation pressure
- Distance from the total internal torque to c.g.
- Moment of inertia, mass xx
- Moment of inertia, mass xy
- Moment of inertia, mass xz
- Moment of inertia, mass yy
- Moment of inertia, mass yz
- Panel area
- Total volume

Another important product of this subsystem is a CAD drawing that provides information on the placement of each component. SolidWorks drawings of the selected architecture can be found in Appendix A.

Assumptions

The limit of the coefficient of drag on a blunt body in the upper atmosphere, computed using free-molecule flow, is found to be 2.0.[9] Since extensive modeling would be required in order to produce a more accurate number, this value is used as a constant throughout the design. A sensitivity analysis should have been performed on this value, but was not due to time considerations.

Another important assumption is that the antennae can be folded along the main bus in order for the spacecraft to fit inside the launch vehicle payload fairing.

In order to distribute the mass of the structures and mechanisms (cabling, small struts, etc), it is contained in the mass of the main bus, which is evenly distributed along the length of the bus.

It is also assumed that solar arrays would be able to be attached to the body of the main bus.

Fidelity Assessment and Verification

The fidelity of the parametric outputs is only as accurate as the inputs used to generate them. Since all of the inputs are physical parameters of subsystem components, the fidelity of the model depends on the combined fidelity of all of the contributing subsystems. Some of the

[9] Hoerner, Sighard, Fluid-Dynamic Drag, Sighard Hoerner, 1965

outputs, however, such as the distances, are approximate values, derived from the SolidWorks configuration. These are approximated since those values do not change appreciably during the ICE session iterations, and it is costly time-wise to input these values at each iteration. The SolidWorks model directly reflects the inputs from each of the subsystems.

An electronic copy of the configuration subsystem sheet can be found on the XTOS compact disc (Configuration.xls).

7.4.7 Power and Pyrotechnics

Introduction

The Power and Pyrotechnic Subsystem (Power and Pyro) sheet selects and sizes the components of the electrical power system (EPS) for the spacecraft. The design methodology follows the steps listed in SMAD section 11.4 for both the power generation and energy storage components. A significant portion of the component sizing is carried out automatically based on the average and peak power requirements of the various subsystems. However, the user has the option of making a number of trades that can dramatically reduce (or inflate) the EPS mass and size for a given set of power requirements. These trades include solar array materials, power regulation schemes, solar array configuration (partly constrained by spacecraft configuration), battery couple, discharges per orbit, depth of discharge, and redundant components. In addition to these trades, the sheet also features variable degradation factors that allow the user to adjust the conservativeness of the design.

Inputs

The design of the EPS is affected by 135 system-level parameters. Most aspects of the mission affect the EPS simply because so many other subsystems have specific power needs. The primary drivers in the size, mass, and complexity of the EPS are as follows:

- Mission lifetime
- Spacecraft configuration
- Orbit characteristics (time in eclipse, etc…)
- Average power loads (per subsystem)
- Peak power loads (per subsystem)
- Bus voltages required (per subsystem)

Outputs

The Power and Pyro sheet passes 95 different parameters as outputs to the various subsystems. Among the largest factors in determining the overall size, mass, and cost of the spacecraft are as follows:

- Solar array mass and size
- Solar array configuration (# of panels, body-mounted vs. deployed, etc…)
- Solar array type
- Solar array power BOL/EOL
- Secondary/Primary battery mass, size, and quantity
- Secondary/Primary battery couples
- Secondary/Primary battery power capacity
- Power regulation and control mass

Assumptions

While the Power and Pyro sheet is considerably robust to various mission types and spacecraft configurations, the level and scope of the design requires that some assumptions be made to simplify the design process.

- The mission consists of only a single satellite
- Solar arrays are the predominant option for a power source.
- The solar arrays are rectangular panels, regardless of configuration.
- The battery dimensions follow a 2:1:1 ratio (length:width:height).
- Transmit power may be needed at any point during daylight or eclipse.

Fidelity Assessment

The fidelity of the Power and Pyro sheet is primarily determined by the accuracy of the information within the solar material and battery couple databases. The attributes associated with each component, such as energy density for solar materials and maximum cycle life for battery couples, play a key role in determining the overall size and mass of the EPS. Because these characteristics vary considerably between manufacturers and over time, the values in the database can be considered conservative averages at best.

The sheet also lacks fidelity in the calculation of mass and power estimates for the power regulation and control equipment (PCU). The mass of the PCU and regulators is estimated using a simple approximation from SMAD that relates total mass to the amount of power regulated. While this is a very rough approximation, a lack of better information exists without actually designing the spacecraft bus and power control systems.

As a final note, the overall conservativeness of the EPS sizing calculations remains in question. Interestingly enough, the size of the body-mounted solar panels in each iteration never actually approached a total area that would fit the spacecraft. While this discrepancy may at first seem like an obvious design conflict, it is not certain whether the solar array size is a product of overzealous power requirement estimates or a product of far too conservative efficiency calculations. Throughout the design iteration process, the power requirements were repeatedly noted as a bit high for such a small satellite. Unfortunately, the level of detail of the preliminary design is not sufficient to determine more accurate power figures.

Verification

Verification of the Power and Pyro sheet was conducted using two methods. First, sample requirements from satellite examples given in SMAD were fed through the sheet to verify that the design results matched (within a small percentage) the results listed. Once the nominal verification had been completed, several inputs were modified to ensure that moderate increases in design requirements yielded only moderate increases in EPS mass and size.

7.4.8 Structures and Mechanisms

Introduction

This subsystem module estimates the vibrational environment that the satellite will experience on the launch vehicle, determines the number of mechanisms required for operation, estimates the power required by the mechanisms, estimates the required structural mass based on a factor of safety of 1.25, and also estimates the launch carrier mass.

The vibrational environment data consists of sound pressure levels, the acoustic environment, random vibrational environment, the pyrotechnic shock environment, peak and sustained accelerations, and the power spectral density. From this data, the calculated natural frequencies of the structure are compared to the frequencies that the satellite may experience to verify that at least the first natural frequency will not be encountered while attached to the launch vehicle.

The number of mechanisms on board the spacecraft depends on the type of power source and the type of antenna. Depending on the combination of power source and antenna, the number of mechanisms required for satellite operation is determined as well as the power required.

The structural mass required is calculated based on the mass of the subsystem components, payload mass, and external component masses. Based on the selection of the primary structural material, the loads (axial and lateral) and stresses are calculated based on the spacecraft structural dimensions.

Inputs

- The structures and mechanisms subsystem uses the following inputs:
- Launch vehicle vibration data
- Subsystem component masses
- Payload mass
- External component masses (solar arrays, antennas)

Outputs

- The structures and mechanisms subsystem outputs the following:
- Average and peak power requirements
- Structural mass
- Launch carrier mass
- Structural reliability

Assumptions

This model makes use of the mass of the satellite subsystem components to estimate the structural mass required. The estimate of structural and cabling mass is based on percentages of the subsystem component masses. The estimate of the launch carrier mass is based on a percentage of the satellite mass including fuel and contingency. Each structural component is to be designed to have a reliability of 99.999%, or such that the overall structural reliability is greater than 99%.

Rationale for simplification

The use of satellite subsystem component masses for estimation of the structural mass is a good approximation in the preliminary design phase. The actual design of the structural truss and launch carrier is a detail design issue and would be nearly impossible to construct based on the dynamic state of this design tool.

Fidelity Assessment

The properties of the materials available for the structure and launch carrier material are that of well-known and documented materials. The percentage estimates of structural mass based on subsystem components are rule of thumb estimates and have an error associated with them based on the truss arrangement in the detail design phase. The vibrational and shock environments of the launch vehicle that the satellite will experience are estimated based on available launch vehicle performance data. This data is the most accurate that could be found, which may include some rounding error associated with the actual environment. The power required (peak and average) for the operation of the spacecraft mechanisms is an estimate that is dependent on the type of mechanisms used and will vary depending on the inertia of the actuated component.

Verification

The structures and mechanisms subsystem module was tested under various launch vehicles, structural material, and design parameter changes. Under all the tested conditions, the structure was not subjected to frequencies above its first natural and the structural dimensions were scaled appropriately to ensure a factor of safety of 1.25. All structural and stress calculations were based on solid mechanics equations and were verified.

7.4.9 Command Control and Data Management (CCDM)

Introduction

The Command Control and Data Management subsystem is responsible for the RF communications link and all the avionics. The system is divided into two primary segments. The Telecommunications segment manages the RF link and all associated hardware. The C&DH segment contains all the avionics, software and the data recorders.

Telecommunications

The Telecommunications segment is comprised of two Low-Gain Antennae (LGA) assemblies. Each assembly contains:

1 Conical Log-Spiral Antenna
1 Multiplexer
2 Filters (Bandpass)
1 Transmitter
1 Receiver
1 I/F Amplifier (IFA)
1 Low Noise Amplifier (LNA)
1 High Power Amplifier (HPA)
1 Automatic Gain Control (AGC)
1 Crystal Oscillator (XO)
2 Mixers
1 Set of cabling[10]

The conical log-spiral antennae allow a 0 dB gain with greater than 360o 3dB beam-width. This means the antenna does not intrinsically introduce any signal loss. A standard low-gain spiral antenna will have a negative gain requiring larger amplifiers. The X-TOS LGAs each have a 270o 3dB beam-width, as seen in Figure 7–2.

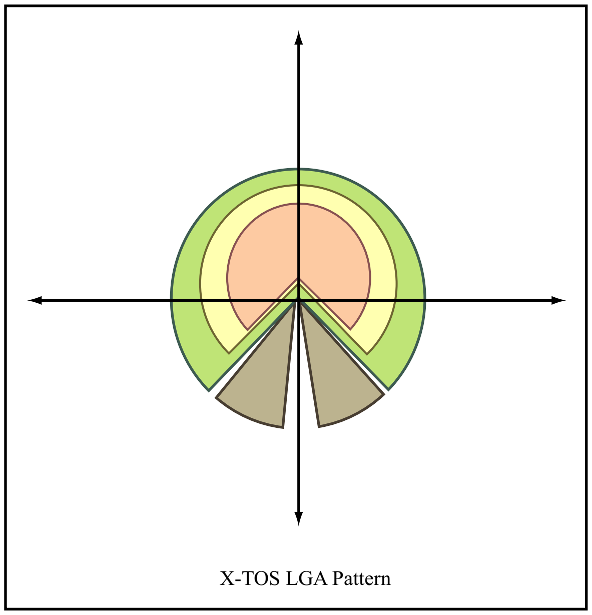


Image by MIT OpenCourseWare.

Figure 7–2: Low Gain Antenna Pattern

[10] This was accounted for within the structures subsystem but is mentioned here for completeness

This was selected to allow for maximum overlap of the two LGAs without the spacecraft body impinging on the signal. Each antenna is made of an aluminum wire wrapped around a plastic cone. The cone is 0.1 m at the base and 0.33 m high. The spacing between the turns in the wire is determined by the frequency. We have chosen a nominal S-Band frequency of 2.2 GHz on the uplink and 2.5 GHz on the downlink. These frequencies are compatible with both AFSCN and TDRSS. These frequencies determine a spacing of 0.068 m between turns. Each cone is mounted on a 0.25 m long, 0.06 m wide plastic cylinder to bring the antenna pattern far enough from the body of the spacecraft.

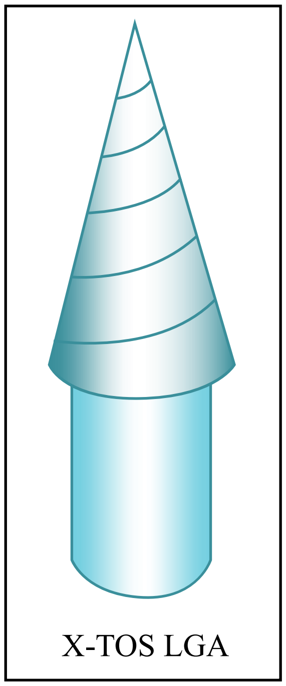


Image by MIT OpenCourseWare.

Figure 7–3: Low Gain Antenna

The antenna is connected to the accompanying hardware by coaxial cables with BNC connectors at either end. The line loss for these connectors was assumed at a length of 1 m (5dB). This is a conservative estimate that is justified because the exact location of the hardware box within the spacecraft bus is not yet known. Ideally, the hardware box should be at the base of the spacecraft cylinder. The uplink will pass from the antenna to the LNA, through the multiplexer, and on to the mixer where it is combined with the signal from XO and down-converted. After that, it will be filtered and passed through the receive IFA. It will then go through the receiver which will demodulate the data and finally be routed to the appropriate On-Board Processor.

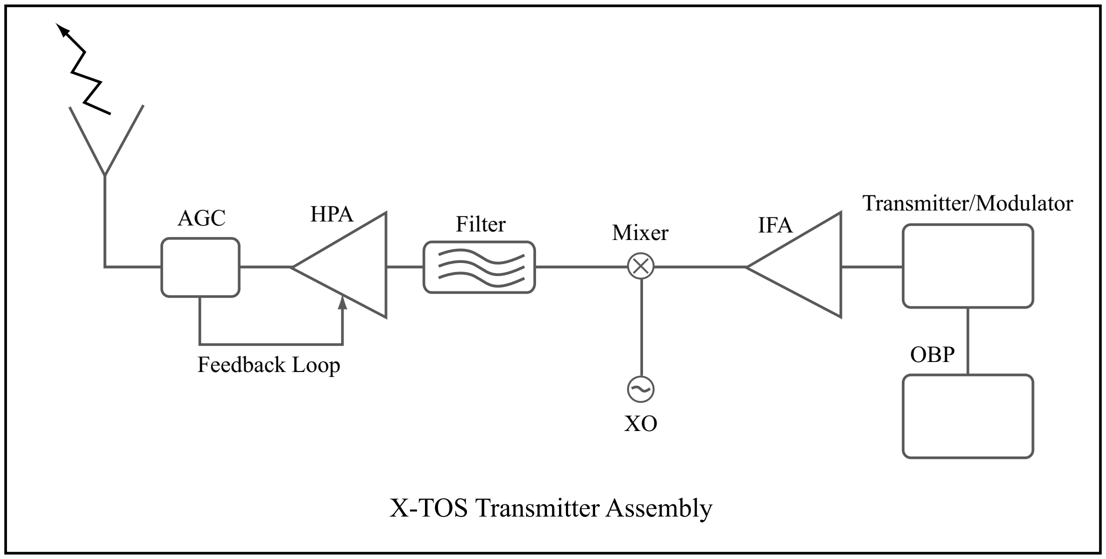


Image by MIT OpenCourseWare.

Figure 7–4: Transmitter Assembly

The structure of the command will be such that it will contain sufficient routing information for the subsystems within the spacecraft bus. The downlink will be generated by the On-Board Processors routed through the transmitter for modulation and through the other IFA. Then it will go through the mixer where it will be combined with the signal from the XO and up-converted. It will then pass through the multiplexer and HPA. The signal will then pass through the AGC and be transmitted out the antenna. The AGC will pass a gain regulation voltage back to the HPA.

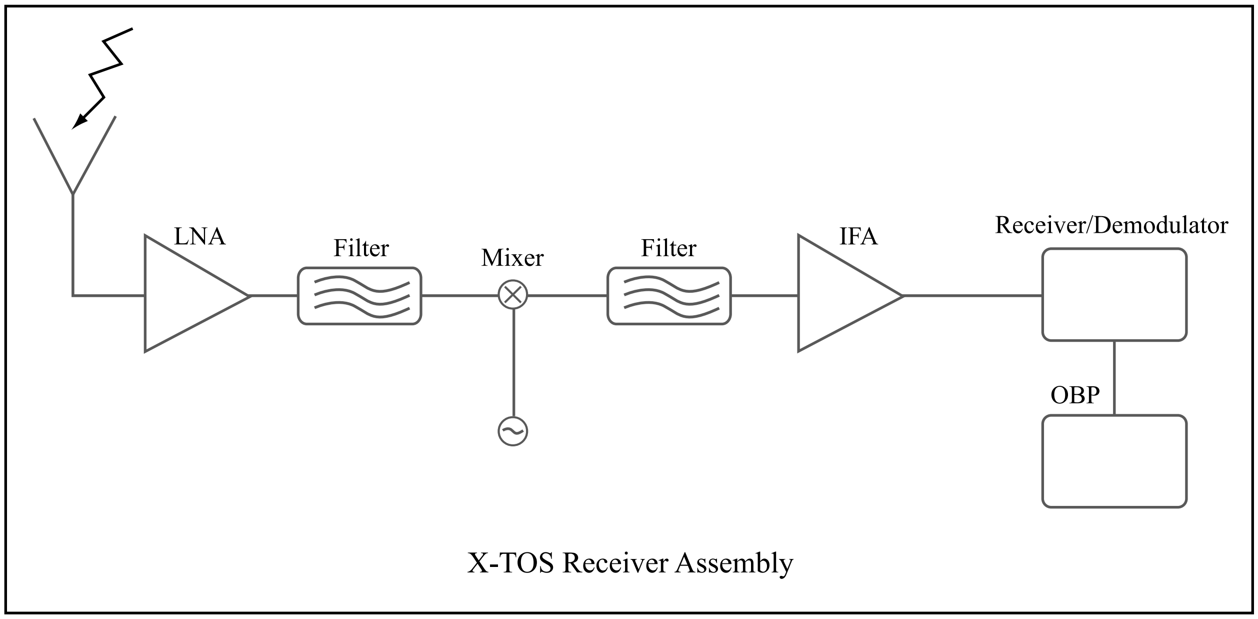


Image by MIT OpenCourseWare.

Figure 7–5: Receiver Assembly

All connections should be made with standard 50 ohm coaxial cable using BNC connectors. The VSWR of the cabling should be no greater than 2:1. Though the design states X-TOS can, from a protocol standpoint, communicate with TDRSS the system as is does not have enough power to talk to such a small aperture dish as TDRSS. Unless the power is significantly increased we must use the high gain (100-200 dB) dishes of the AFSCN. The telecommunications sub system was designed to a 4dB link margin.

Command and Data Handling

The C&DH system contains all the avionics and manages all the software. The hardware is as follows:

2 General Dynamics 4063RT On-Board Processors (OBP)
1 General Dynamics 4063RT Contingency Processor (CP)
5 I/O cards (2 for each OBP 1 for the CP)
2 Digital data bus switch.
2 High speed communications card (For connection to Telecommunications)
2 Aero-Astro S4 20Gbit data recorders
1 Set of cabling

The OBPs and the CP are the same model computer. The OBPs are redundant computers for nominal operations and are in operation for modes 1-6. The OBPs maintain all mission algorithms and contain all necessary boot data on an EEPROM. The OBPs should contain several memory slots for ground generated sequences. The CP is a special computer who's only use is for Safe Mode (mode 7). The CP will be off with the safe mode algorithms stored in an EEPROM and run them on power up. This should ensure the safe mode algorithms are not corrupted by any glitch in the system. The first three I/O cards connect the OBPs and CP to the spacecraft data bus. They should use IEEE standard radiation hardened 50 pin IDE cables. The cables should be connected to the digital switch which will route data to the prime OBP. The digital switches should be connected in parallel and the redundant one should remain off unless the prime one goes down or is commanded off. All the other subsystems should also be connected by 50 pin IDE cables to the digital switch this then comprises the data bus. The digital switch should allow for duplexed routing of data. The CP should also be connected to the digital switch in the same manner as the OBPs. The OBPs should also be connected via the last two I/O cards to the data recorders. Only the prime OBP should transmit data to the recorders. If one recorder is full or brought off-line, data will be routed to the alternate recorder. The OBPs will be connected to the Telecommunications System via the high speed communications cards. The data will be transferred by the prime OBP from the recorder to the telecom system and sent to the ground. The recorders can each hold five orbits worth of data. We should to attempt to nominally dump the data at most once every three orbits in order to both keep the data rate down and be able to downlink it all in one ground pass. The CP should be placed on a different power bus than the OBPs if at all possible. This would allow the CP to react to a short or other power emergencies on the main power bus.

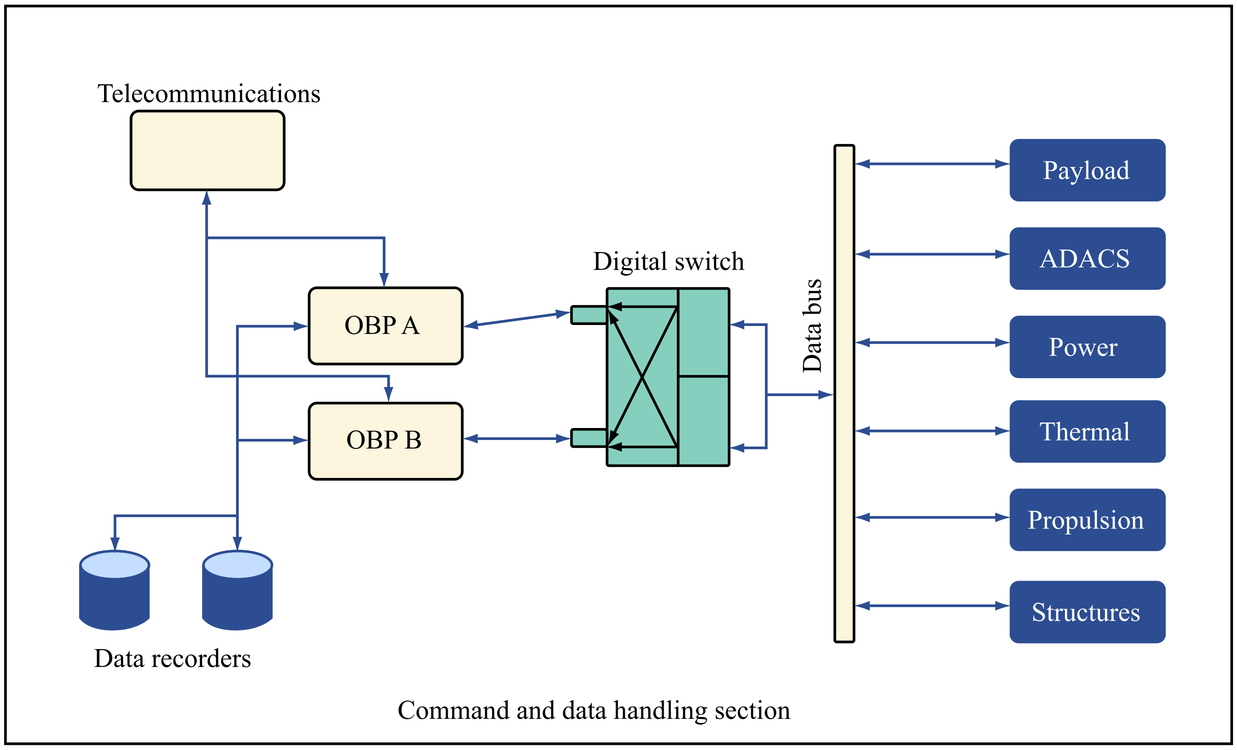


Image by MIT OpenCourseWare.

Figure 7–6: Command and Data Handling Section

Data rate, software and throughput requirements are primarily derived from the input from other subsystems. The raw data rate is 17.901 Kbps. That data rate is multiplied by 16 allowing 15:1 ratio of dump data to real-time sent on the downlink. To achieve a BER of 10-7 with a 7dB coding gain the data is Reed-Solomon encoded using a (255,223) code then Verterbi encoded. This means that first in each code word there are 223 8-bit information symbols and 32 8-bit error correction symbols. The code used is AFSCN, TDRSS and CCSDS compatible. The data should then be Verterbi encoded as the second phase of error correction coding. This code should have a rate of ½ and a constraint length of 7. This means that seven bits are modulo-2 added to produce two bits, The bits are shifted by one and the process is repeated. This process results in a combined total downlink rate of 664.502 Kbps.

The software needed by the system is estimated by adding the software requirements of all the subsystems and adding a 35% margin. The total resultant code size is 900 Kwds where a word is 8 bits. The total throughput of the system is estimated in a similar fashion and is 22.3 MIPS. The hardware discussed in the previous paragraphs was sized to meet these requirements.

CCDM Subsystem Trades

No CCDM subsystem trades were performed within an ICE session. These were examined off-line in order to verify this design was the most ideal one. The trade study only made certain qualitative assessments of the impact on both configuration and structures. The other options explored for the telecommunications system were communications with TDRSS and a high gain antenna. These selections decreased the overall utility of the system in the high drag environment. TDRSS with a low gain antenna required a significant power increase and even though it allowed for longer possible communications time would have driven power requirements too high for a small satellite mission. AFSCN with a high gain antenna does not allow for a long enough pass to downlink significant amounts of data. And while TDRSS with a high gain antenna is fine from both a power and downlink perspective. A 1.9 m dish on the side of the spacecraft in a high drag environment provides a large drag force. In addition The mass of an aluminum dish and all the actuating equipment is much greater than the two plastic, wire coated antennae we currently have.

ICE Sheets

The ICE sheets for the CCDM subsystem consisted of two sets: one for Telecommunications and one for C&DH. The telecom sheets took as input each of the spacecraft modes and their duration, the overall data rates compiled by C&DH, the required link margin and the orbital data. The C&DH sheets took all the data rates code and throughput requirements from all the subsystems. It also took spacecraft mode and orbit information. The telecom sheets outputted the antenna design, link performance, mass and power requirements, a maximum downlink time per orbit and a data latency figure for utility analysis. The C&DH sheets outputted the volume and mass for each component, the mass and power for the system as a whole. It also publishes the available and required code size, the temperature output of the system, the data storage capacity, the total data rate and the encoded data rate. No functional verification or fidelity assessment was performed on these sheets independently.[SML1]

7.4.10 Thermal

Introduction

The thermal sheet accepts inputs from nearly all sub-system sheets, in order to set a maximum and minimum operating temperature for the spacecraft. With these constraints in place, the user can choose two outer materials for the spacecraft. These materials include solar panels, several different types of metal and non-metal surfaces, and a variety of painted surfaces. By choosing these surfaces accordingly, the user designs a thermally balanced satellite. After this balance is achieved, the sheet calculates the mass and power required to insulate and heat the fuel tanks and lines.

Inputs

The thermal subsystem uses the following inputs:

- Maximum and minimum operating temperatures for different spacecraft systems
- Summed power requirements for power dissipation

Outputs

The thermal subsystem outputs the following:

- Mass and power summaries for the thermal system
- Maximum and minimum allowable operating temperatures

Assumptions

The biggest assumption in the baseline model (inherited from C-TOS) is the idea of the best case/worst case scenario. The sheet calculates the equilibrium temperature for the spacecraft in full sunlight, as well as the temperature of the spacecraft in full eclipse. This assumes an isothermal satellite. For the C-TOS sheets, this means that the spacecraft would require expensive active thermal control systems in order to counteract these scenarios. For the X-TOS satellite, it is known that the orbital periods will be on the order of 90 minutes, and therefore the satellite will never actually reach such extreme thermal equilibriums. Instead, a simple dynamic thermal calculation is run to show, based on the spacecraft's mass, how the temperature will change over time. This allows the user to design a thermally balanced spacecraft that stays within the temperature constraints.

Fidelity Assessment

The single point of glaring infidelity inherited from C-TOS is the aforementioned isothermal assumption. Even in a small and simple satellite, there are obvious locations (near instrument packages, etc.) where the local temperatures will be different than those of a general satellite. The model fidelity is increased somewhat by running two separate thermal balance calculations—one for the satellite in general, and one for the ram face of the satellite, which contains two of the (fairly high powered) instruments. However, at this level of fidelity, we make no provisions for the transfer of heat, which would almost certainly be required for the final design to remain in thermal balance as predicted.

Verification

The outputs of the model seem to agree with intuition—by choosing surface materials that complement one another, one can build a satellite whose heat surplus in one case is almost exactly balanced by its heat deficit in another. The mass of the satellite (> 100 kg) means that the rate of thermal change in the two regimes will be very slow, as shown by the dynamic model tested. It has also been noted that the vast majority of short lifetime, small satellites have passive thermal control systems.

7.4.11 Attitude Determination and Control System (ADACS)

Introduction

The ADACS subsystem is in charge of the attitude determination and attitude control of the spacecraft. It estimates all the disturbances that the satellite will experience in the upper-atmospheric orbit, and given the pointing accuracy needed, will determine what precise ADACS system to use and how much fuel will be needed to control the satellite's attitude.

The disturbances that the satellite will experience are:

- Aerodynamic disturbances
- Gravity gradient torques
- Solar pressure
- Internal torques (which in this case are negligible because the satellite does not have deployable solar panels or other moving parts).

Once these disturbances are calculated, he choice of the most appropriate ADACS sensors and effectors is made as a consequence of the payload and communication subsystems pointing requirements. The total mass and power required for the selected ADACS equipment is output. An estimation of the Delta V per orbit for the attitude control is also calculated based on the disturbances, the pointing control requirements, and the type of ADACS effectors chosen.

Inputs

The ADACS subsystem primarily uses the following inputs:

- Total lifetime
- Altitude of the orbit
- Momentums of inertia of the spacecraft
- Type of thrusters (Isp, Thrust)
- Pointing requirements

Outputs

The ADACS subsystem outputs the following:

- Average and peak power requirements
- Size and mass of the ADACS components
- Delta V per orbit for the attitude control

Assumptions

The main assumption is that the center of gravity of the spacecraft is in front of the center of aerodynamic pressure. This assumption results in aerodynamic stabilization of the satellite and thus eliminates the aerodynamic disturbances from the calculation of the Delta V needed for altitude control. A more precise study of the aerodynamic stability of the spacecraft would be required to determine the additional modifications needed to achieve such stabilization.

The other assumption is that the thruster clusters used for propulsion can also be used for the attitude control as ADACS effectors.

Fidelity Assessment

The properties of the ADACS sensors (power required, mass and pointing accuracy) are well known and very precise. However, the reliability and lifetime of these instruments is not documented due to their short history.

Computationally, the calculations of the gravity gradient and solar pressure disturbances are based on precise astrophysics calculations, and the calculation of the aerodynamic disturbances does not have an influence on the ADACS subsystem calculations (once the assumption of an aerodynamically stabilized spacecraft is made).

7.4.12 Propulsion

Introduction

The Propulsion Subsystem Sheet was based extensively on the work of the C-TOS team. However, major modifications were made to incorporate the "station-keeping" thruster concept (see below for detailed explanation). The basic function of the sheet is to size both the ADACS and station-keeping thrusters and then calculate the total fuel mass required for the mission.

Originally designed for C-TOS, the sheet sizes a number of "thruster clusters" based on inputs from the ADACS sheet. Based on the assumption that the spacecraft will be 3-axis stabilized, each cluster can consist of up to four thrusters for attitude control. The rear-facing thruster will therefore be used periodically to modify the orbit via a station-keeping maneuver.

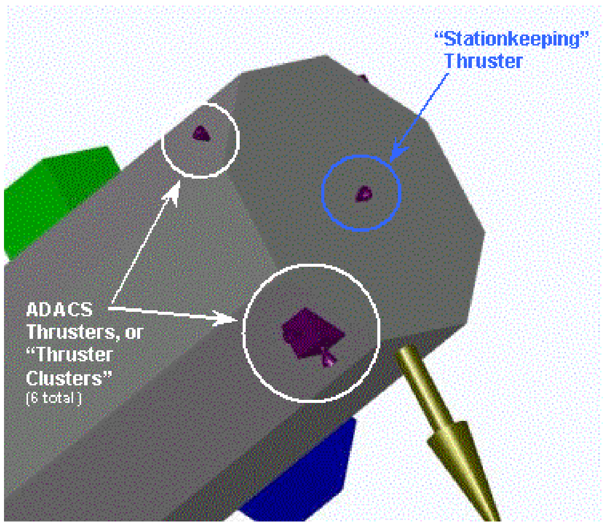

"Stationkeeping" Thruster
ADACS Thrusters, or "Thruster Clusters" (6 total)

Figure 7–7: Thruster Locations

After examining the unique mission of the X-TOS spacecraft, it is evident that far fewer thrusters will be required to adequately perform the X-TOS mission compared to the C-TOS mission. Since the spacecraft will always be oriented in the same direction relative to its velocity vector, true three-axis control is no longer necessary. In addition, the unique, high-drag environment that will be encountered will partially help maintain the orientation of the vehicle once it is aligned.

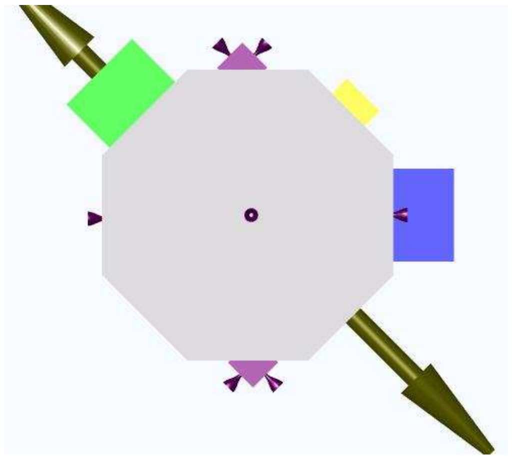


Figure 7–8: ADACS Thruster Configuration

Upon calculating the maximum drag, it was noted that a small (< 5 N) thruster could be placed at the rear of the spacecraft in order to maintain the desired altitude throughout the mission lifetime. In order to minimize mass and complexity, it was eventually decided that this single stationkeeping thruster should share fuel, tanks and other components with the ADACS thrusters.

Besides electric propulsion, other alternative methods of propulsion were only explored in discussion. Once it was determined that the fuel mass would be a significant driver in the overall mass of the satellite and the ultimate determination of mission lifetime, a high Isp was obviously desired. This ruled out cold gas systems. The next propulsion type considered was bipropellant. Although it afforded some improvement in Isp over a monopropellant, hydrazine system, quick “Rough Order of Magnitude” calculations which were based on the engine data contained in the C-TOS sheet showed that in the range of our spacecraft (100 to 400 kg containing 40 to 200 kg of fuel), the additional mass of isolated storage tanks, additional valves, tubes and regulators would essentially cancel out the decrease in mass afforded by the higher specific impulse. In addition, there were more failure modes and expense associated with the bipropellant system.

Returning to the electric propulsion issue, the sheet was set up so that the X-TOS spacecraft could be equipped with electric propulsion. It was noted, however through consultations with the Power and Pyro Chair, that the power required to drive these thrusters would significantly alter the requirements for the spacecraft in terms of solar panels and power storage (batteries). Thus, explicit trades were never performed using electric propulsion as an option.

In retrospect, the decisions to eliminate cold gas, electric and more importantly, bipropellant propulsion systems may have been made prematurely. Part of the power of the MATE and ICE processes lies in expanding the normal tradespace quickly and easily through parametric models. Often times, the assumptions we made at one stage in the process were proven wrong for our particular mission. Every engineer has longstanding beliefs about the interactions of certain subsystems, however in order to rigorously explore all of the possible design trades using MATE and ICE, these assumptions have to be suspended until a more accurate understanding of the unique aspects of the current system can be gained.

Inputs

The key Inputs to the X-TOS Propulsion Subsystem Sheet are:

- Total Mission Delta-V (“Delta V, total”)
- Delta-V per orbit for attitude control (“Delta V - ADACS, per Orbit”)
- Total Wet Mass of the Spacecraft (“Mass, total wet - w/ cont”)

Outputs

The key Outputs of the X-TOS Propulsion Subsystem Sheet are:

- Fuel Tank Size (“Dimension 1 – Fuel Tank, Dimension 2 – Fuel Tank”)
- Mass of the Fuel Tank (“Mass - Fuel tank”)
- Total Mass of Fuel (“Mass, loaded – Fuel”)
- Total Mass of Pressurant (“Mass, loaded – Pressurant”)
- Mass of Fuel available for Stationkeeping and ADACS (“Mass, Propellant - Stationkeeping + ADACS”)
- Total Mass of All Propulsion Components (“Mass, total dry – Propulsion”)
- Average Power (“Power, average - Propulsion (mode 2)” etc)
- Peak Power (“Power, peak - Propulsion (mode 2)” etc)
- Thruster Exhaust Products (“Products, exhaust - Stationkeeping Thruster” etc)
- Specific Impulse (“Specific impulse, min - Stationkeeping Thruster” etc)
- Propulsion System Type (“Type - Stationkeeping Thruster” etc)
- Mass of Fuel required per orbit for ADACS (“Mass, Propellant - Integrated ADACS, per Orbit”)

Assumptions

Some key assumptions drive the ultimate size and mass of the propellant tanks. The calculations of the mass and volume of the Helium pressurant are based on the Beattie-Bridgeman equation. This is carried over from the C-TOS model. In addition, there are several ‘holes’ in the engine database (for operating power, mass and dimensions) which have to be estimated in order to perform some trades. These do not apply to the final engine choices.

Fidelity Assessment

The most important output of the Propulsion sheet, the mass of propellant, is based on the rocket equation. This has obviously been proven to be highly accurate over the last 60 years. In addition, the final choice of thrusters (available “off the shelf”), affords us a high degree of reliability in our Isp and mass estimates. However, there are several other aspects of the sheet that can add error to the final outputs. There are some miscellaneous inputs for additional valves and mounting provisions, most merely guesses, but they account for only a small percentage of the final propulsion system mass. The size and mass of the fuel tank itself is also subject to some estimation error, but the PV/W method is based on historical data and should be accurate to within a few percent (see references on sheet).

Verification

The only real test of the sheet’s output came through a true, system-level design session. Upon outputting the Isp (so that the Mission sheet could calculate the life of the spacecraft based on the periodic depletion of fuel over each orbit), it was found that the C-TOS engine database actually contained nozzle exit velocities, not specific impulse data. This discrepancy was found by examining how reasonable the calculated lifetimes were based on the fuel mass carried on a particular design. Although this was not the most technical approach, it was another example of the absolute necessity of having experienced participants in the process.

7.4.13 Cost

Introduction

This module used two Cost-Engineering Relationship (CER) models to determine the cost of the spacecraft, launch, and associated upfront operations. The first of these was based on Space Mission Analysis and Design 3rd edition (SMAD) for typical spacecraft missions weighing between 253 and 1153 kilograms (dry). The second model was also from SMAD, and was based on Earth-orbiting small satellites, weighing between 20 and 400 kilograms (dry). Both are parametric models based on historical information.

Inputs

There were 43 inputs in the module, primarily focused on mass and power from the various subsystems. The following list is of the major drivers in the cost.

- Spacecraft Bus Dry Mass
- Structures Mass
- Thermal Mass
- Thermal Average Power
- Power Subsystem Mass
- Solar Array Area
- Battery Capacity
- BOL / EOL Power
- Telemetry, Tracking & Command and Command & Data Handling Mass
- Downlink Data Rate
- Data Storage Capability
- ADCS Dry Weight
- Pointing Knowledge
- Number of Thrusters
- Launch Cost

Outputs

Again, there were a total of 26 outputs, or approximately 13 outputs for each CER. They included the following costs:

- ADACS and Propulsion
- Telemetry, Tracking & Command and Command & Data Handling
- Integration, Assembly, and Test Wraps
- Program Level Wraps
- Ground Support Equipment Wraps
- Launch & Orbital Operations Wraps
- Power and Pyro
- Structures
- Thermal
- Spacecraft
- Wraps

…as well as total initial program costs (not including extended operations).

Assumptions

The Cost-Engineering Relationships are based on historical data, and hence have ranges for which they are valid. Those are not included in this section.

- The mission will consist of only one satellite.
- There is a single launch.
- Each subsystem accounts for multiple components--the Cost Module receives the totals.
- The payload is furnished by the customer, and hence is not considered part of the cost.
- Extended mission operations are not considered due to lack of knowledge of the staffing and resources to be used.

Description

The Cost-Engineering Relationships used in this module, and available through SMAD, are based on previous missions, where subsystem information is available, and costs are known. From this data, parametric models are created, whereby relationships between known engineering values are correlated to costs. This module uses two common cost models, to provide a comparison, and estimate rough costs.

Fidelity Assessment

The fidelity of Cost-Engineering Relationships in general is questionable, and at best can be used to estimate a rough cost. In fact, error margins on these relationships can be on the same order of magnitude as the estimations themselves. Furthermore, these are based on historical data, and a very small sample set. This means that they fail account for changes in the field, such as falling costs of supplies and technology, the use of uniform spacecraft buses, and so on. As a result, they should be considered only as rough estimations, and provide relative order of magnitudes to compare spacecraft with one another.

Verification

Several methods were used to verify the outputs of the cost module. The first, and most notable one was to ensure that all values inputted were within the acceptable data range for the CERs. Secondly, two CERs were used to correlate information. It was expected that the first model, using all typical spacecraft types would yield a larger cost estimation than the other, which was based only on small spacecraft. This held true for the various spacecraft run through the ICE process. Thirdly, two outside models were used to verify the order of magnitude on the SMAD modules. The first of these was the Aerospace Small Satellite Cost Model, which is based on recent, smaller spacecraft. The second was NASA’s Space Operations Cost Model, which was used to verify that for short lifetimes (less than one year) the cost of operations was negligible compared to the program cost. Finally, operator intuition was used to make sure that the outputted numbers seemed accurate.

7.4.14 Reliability

Introduction

This module uses Markov Modeling to determine the probability of the XTOS mission being in any given operational state (full functionality, one or two failed instruments, or system failure). The probabilities are given at design mission lifetime. The module is implemented in Excel/ ICEMaker, but a Matlab function is called to perform the calculations.

Inputs

- The reliability subsystem/module uses the following inputs:
- Mission lifetime
- Failure probability of launch vehicle
- Failure rate of the instruments
- Failure rates of the different subsystems (ADACS, C&DH, Power & Pyro, Propulsion, Structure & Mechanisms, Communication, and Thermal Control)
- Number of replicates for the different subsystems (ADACS, C&DH, Power & Pyro, Propulsion, Structure & Mechanisms, Communication, and Thermal Control)

Outputs

- The reliability subsystem/module outputs the following:
- Probability of achieving target life
- Probability of achieving lifetime with full functionality
- Probability of achieving lifetime with one failed (secondary) instrument
- Probability of achieving lifetime with two failed (secondary) instruments

Assumptions

- The mission will use a single spacecraft.
- Only one launch is permitted, therefore removing the possibility for repair or replenishment.
- Only one set of instruments is available (no redundancy).
- Instrument 1 is mission critical.
- Instruments 2 and 3 are identical as far as reliability and criticality are concerned.
- The failure rates for the different subsystems are constant over time.
- The spacecraft fails if any subsystem fails.
- The spacecraft and its subsystems can only be in a failed or functional state.
- The replicates for each subsystem are placed in parallel.
- The replicates for each subsystem have the same failure rate (but not necessarily the same design).

Description

The reliability module uses a Markov Model to determine the probability of the mission being in any given state as a function of time during the mission. For the cases considered here, the module calculates reliability information at the end of the mission life period, but it has the capability for providing the reliability at any other time.

The fault tree considered to compute the different probabilities is the following:

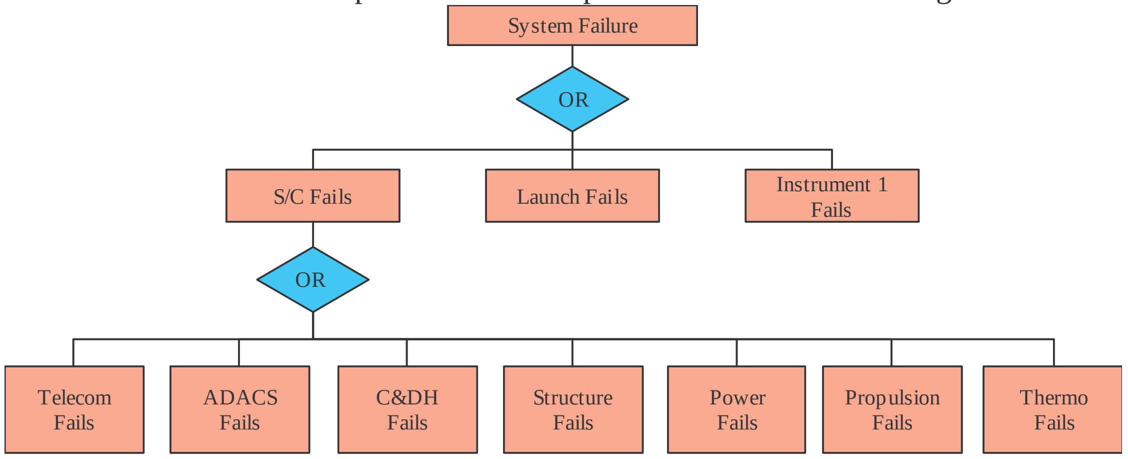


Figure 7–9: Fault Tree

Based on this fault tree, the following states are defined:

- State 1: Full functionality
- State 2: Instrument 2 or Instrument 3 fails
- State 3: Instrument 2 and Instrument 3 fail
- State 4: System failure: Launch or spacecraft or instrument 1 failure

which leads to the following state diagram:

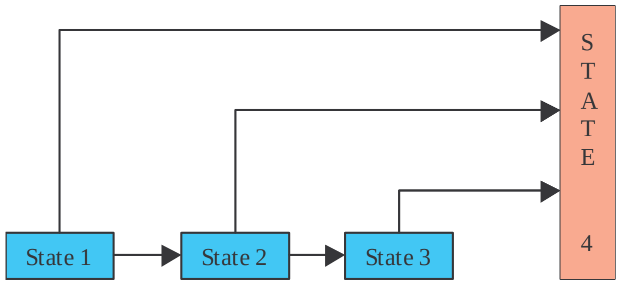


Figure 7–10: State Diagram

Using the subsystems failure rates (inverse of the Mean Time Between Failures (MTBF)) and number of replicates, the module first calculates the spacecraft reliability and MTBF. Exponential models are used for all reliabilities: $R = e^{-ft}$, where f is the failure rate and t is time.

The code then uses the state diagram, the spacecraft failure rate and the instruments' failure rates to calculate the Transition Matrix A, defined by: $P = AP$, where P is the state vector:

$P = [P_1 \quad P_2 \quad P_3 \quad P_4]^T, P_i = P(State\ i)$

Note that, because launch is a single event with a probability of failure (as opposed to a MTBF), its effect is not included in the Markov model, but is added to the final state vector. The Transition Matrix and the lifetime are then used to compute the different probabilities at mission lifetime. The launch probability of failure is incorporated at the end to give the final outputs.

Fidelity Assessment

The fidelity of the reliability module suffers mostly from a lack of knowledge about the true mean time between failures of the various subsystems and instruments. Representative numbers are used, but eventually these numbers will need to be improved based on typical values used in industry.

The different subsystems should also be further refined for a more precise computation of their respective reliabilities. Because true numbers are not available, the subsystems are considered black boxes, and redundancy is applied to the subsystem as a whole. This simplification should not significantly affect the results, though, because non-redundant parts are usually highly reliable.

Verification

The reliability module was tested using various combinations of initial parameters, including a variety of mean time to failures, number of replicates for each subsystem, and mission life times. Realistic outputs were sought.

7.5 Design Convergence

While all the subsystems seem to operate as direct feed-through models, the aggregate input/output dependencies of the ICE subsystems can create semi-implicit loops. As mentioned earlier, there is a strong interdependency among spacecraft subsystems and their Excel workbooks. Therefore, the publishing of a changed Output must propagate through all the subsystems several times before a design is said to have converged. The term convergence, in this context, refers to the stabilization of all propagating parameters to within five percent of the mean value in three consecutive updates.

7.6 Mapping Converged Design to Utility

The Multi-Attribute Tradespace Exploration (MATE) process generated a series of satellite architectures and their respective customer utility. In the Concurrent Engineering (CON) portion of the design process, the chosen architecture was point designed with the ICEMaker software as a baseline design with a utility value known a priori. After the baseline design converged, subsystems were able to conduct design trades in an effort to increase the customer utility. Once the traded design had converged, the satellite architecture was run back through the utility function to generate a utility value for the new design. This in effect, was mapping the converged design back to the utility function in an effort to provide the customer with a better product.

7.6.1 MATE Verification

The concept of trading subsystem parameters in an effort to provide better customer utility is an integral part of the Concurrent Engineering design process. In order to determine the effect of the subsystem trade on utility, it was necessary to re-run the converged satellite architecture through the utility function created in the MATE phase of the design process. A MATE specialist monitored this process. The benefit of monitoring this subsystem trade versus utility gave a quantitative assessment of the value and direction of the trade. The design team was therefore able to trade for a design that would provide the highest utility.

7.6.2 Multiple Utility Functions

The MATE-CON process depends heavily on having the customer(s) monitor the design process. When defining the mission and enumerating the tradespace, there is usually a primal utility function that all of the architectures are associated with. The point design architecture, usually a combination of highest utility and lowest cost, is then designed to the known utility in the concurrent design phase. Subsystem trades are then made in an attempt to increase the utility, which is monitored by the MATE specialist. In any phase of the design process, the customer may change his mission preferences through a follow-up interview which reflects a change in the utility function. Depending on the phase of the design process, a re-enumeration of the tradespace with the new utility function may be practical, and therefore will yield a different point design architecture. If the concurrent design phase has become very involved, the most time efficient method may be to run the current architecture through the new utility function and make subsystem trades in an effort to increase the new customer utility. Once again, the MATE specialist is able to determine the direction of the subsystem trades. Multiple utility functions, or multiple interviews with the customer, are instrumental in ensuring that the designers/contractors provide the highest quality product.

7.7 Lessons Learned

7.7.1 ICE Process

The final version of our ICE sheets represented a strong, parametric satellite model. In general, each member of the class was satisfied with the performance of their own worksheet, but was frustrated at the lack of time remaining to fully explore the power of the tool they had developed.

The ICE process differed significantly from the MATE process in several ways. Having followed the MATE process already, our concurrent engineering efforts began with a fairly deep understanding of the mission goals and key parameters. During this phase, the class devoted its time much more pointedly towards the technical details of the mission rather than on the process involved in attaining that data. This was mainly due to the relatively more mature format of the software architecture. Whereas the class really took the liberty to re-design the entire software architecture as it created its MATE models, the structure of the ICE spreadsheets remained predominantly unchanged from previous years' work.

7.7.2 Technical Challenges

A number of lessons were learned by each of the subsystem chairs as they created, re-worked, piloted, and engineered their way through the ICE process. The particular lessons learned have been documented within the appropriate subsystem sheet descriptions in the previous section.

7.7.3 Re-Use of Software Models

A significant debate occurred within the class as to the benefits of starting directly from the previous class' ICE models (C-TOS in this case), or whether it would have been more productive to begin with blank spreadsheet templates that simply contained the macros for passing variables and possibly a list of potential variables to get started with.

A few of the subsystem chairs felt that they gained an invaluable head start by beginning with the C-TOS sheets. In particular, the Structures and Mission sheets only needed minor modifications. However, most of the other sheets required major changes, and the class felt that they spent a disproportionate amount of their time performing this task. Unfortunately, figuring out the logic of a highly complex Excel spreadsheet can take a significant amount of time. This is due in part to the format of the program (i.e. had to click inside formulas, check for cell numbers and hunt around to find what values those cell numbers represented), but also due to the fact that the equations and assumptions were not rigorously documented within the existing sheets. Due to the parametric nature of many of the sheets, many cells contained complicated IF statements and references to numerous other worksheets. When one change was made, such as deleting a "daughter" satellite from the legacy design, many links were broken resulting in the dreaded "#REF."

Finally, the class found that the sub-architectures of the sheets themselves were not consistent with each other. A primary example of this phenomenon was the inclusion of numerous “over-ride” buttons on the “input (general)” and “output (general)” in several of the subsystem worksheets, but not in others. As the concurrent design sessions unfolded, the presence or absence of these buttons became very important. If an error was encountered, and an estimate value could not be substituted, situations arose where the entire group was idle for a considerable amount of time while one small error was being fixed.

The uniqueness of each sheet could also have been attributed to a general feeling of uncertainty among class members regarding a rigorous approach to parametric modeling. Many 16.89 class members had different opinions concerning the robustness of the sheets. At a fundamental level, the debate centered around the inclusion of model limitations and whether inputs and outputs should be checked to see if they fit the acceptable ranges of the equations they were to be applied to. The level of detail contained within the subsystem sheets should be balanced carefully so all sheets are at the same fidelity.

After a prolonged discussion, the class concurred that the initial presence of the C-TOS sheet greatly accelerated the class’ movement down the ICE learning curve because participants were able to visualize how information would flow and understand the mechanics of passing variables and making detailed design trades. Once that learning occurs, however, the class felt that basing their individual sheets only off of standardized templates would be the best approach. These templates would contain all of the necessary macros, input and output pages, over-ride structures, and legacy design databases (such as detailed lists of launch vehicles and propulsion components).

7.7.4 Process Flow

After the ICE sheets were mature enough to begin running design trades, class members began to see the power of the process. Not only did we find that designs could be quickly and accurately generated, but we had a method of evaluating them against each other and mapping out our trade studies (via our MATE chair). During the integrated design sessions, the class worked extremely well together to address inconsistencies and reason through unexpected results. The projected maps of system mass breakdowns and the basic configuration layout were extremely helpful for all involved.

Some general insights into the ICE process were a result of numerous design sessions:

- The preparation of each team member is crucial to a smooth running session – the loss of one sheet created unacceptable delays. Likewise, every team member needs to be present for the process to flow smoothly.
- Complex interactions between subsystem sheets were not detected by individual debugging efforts. Subgroups needed time to work together to understand how their sheets were interdependent.
- The co-location of all participants within one room was seen as extremely important. Several team members had worked on the C-TOS project, which included team members who worked over the Internet. They all concurred that face-to-face interactions were far

superior in obtaining accurate and fast-paced decisions during the early stages of the process.

- The sequencing of the send/receive process could lead to either long delays or premature assumptions of convergence. The class experimented with several ‘protocols’ regarding the order that certain subsystem sheets synchronized with the ICE server. If a general announcement was made to “send and receive” it usually turned out that some of the sheets would be finished before others and thus their last calculations were not based on the latest information from other sheets. Another method was to have the ICE “conductor” signal individually to each sheet when to send or receive information. This generally led to convergence around a particular design point within four or five iterations, however this could take from 5 to 15 minutes. The class recommended writing some sort of macro to automatically synchronize the data.

8 Requirements

8.1 Introduction

Requirements provide a basis on which to design and test a spacecraft system. They lay out specific traits which the system must possess to satisfy the mission. Requirements are typically broken down into three major categories: Project, Functional and Operational. Because operations are not considered within the scope of this project, the operational requirements are not presented. While the system constraints are usually included with the requirements, the constraints have already been discussed in section 2 and will not be repeated here.

8.2 Generating Requirements

In a traditional space system design environment, the requirements are laid out prior to beginning the design process. For the XTOS project, the MATE process used focuses on trading user "preferences" to achieve the best design for the desired mission. The set of requirements are then generated by examining the selected architecture and mapping the attributes it to spacecraft subsystems. This is a unique aspect of the MATE-CON approach—the requirements are actually determined by the architecture, as opposed to the reverse for typical design processes. The requirements for the XTOS spacecraft, as determined by the final architecture, are contained below.

8.3 The Requirements

8.3.1 Mission Overview

The X-TOS mission will provide data about the drag environment in the low earth atmosphere. This will be accomplished by flying the Air Force Research Laboratory's (AFRL) Atmospheric Density Specification (ADS) instrument suite on a single spacecraft. The spacecraft will make in-situ measurements.

8.3.2 Scope

This document will focus primarily on project level requirements for the X-TOS mission. These requirements include some specifications for particular subsystems and requirements levied on the system by ADS-ER-7E2—the AFRL Experiment Requirement Document for the ADS Sensor suite. Because this design is a preliminary one, the requirements are short and allow for some flexibility in the design.

8.3.3 Project Level Requirements

1. The XTOS mission will comply with all requirements in AFRL document ADS-ER-7E2.
2. The design shall preclude the propagation of errors beyond a particular component interface. A component is defined as a self contained collection of parts which perform the same function.
3. XTOS shall be capable of autonomous operations for a period of at least 5 orbits.

4. Except for the primary structure and power bus, no single failure shall result in mission failure.
5. The XTOS mission will be in polar, low earth circular orbit with a radius of 300 km altitude.
6. The XTOS mission will be launched on a Minotaur launch vehicle. The spacecraft will comply with all safety and interface requirements levied by that launch vehicle.
7. The XTOS mission will have a minimum life span of 6 months.
8. Raw instrument data shall be provided to the user community.

8.3.4 Functional Requirements

ADACS

9. The XTOS spacecraft will feature three-axis stabilized attitude control.
10. The attitude control system will provide ± 2^o pointing knowledge.
11. The XTOS mission will have less than a 1^o drift rate in the roll, pitch, and yaw axes.

PAYLOAD

12. All ADS instruments, with the exception of the accelerometer, shall be positioned on the ram face.
13. The ADS accelerometer will be located within 6 inches of the spacecraft center of gravity.

PROPULSION

14. The XTOS mission will use chemical fuel. The fuel chosen will not contaminate the instruments or solar cells.

THERMAL

15. The thermal control subsystem shall be primarily passive, with active heaters to be used when necessary.

POWER

16. The power and pyrotechnics subsystem will provide all power required by the spacecraft subsystems.
17. The power and pyrotechnics subsystem will generate power through body mounted solar arrays. The arrays will be sized such that there will be a 15% power margin at end of life (EOL).
18. The power and pyrotechnics subsystem shall use rechargeable batteries to provide power during eclipse. The batteries will be sized such that there will be a 15% power margin at end of life.

CCDM

19. XTOS will communicate with a ground communications system similar to the AFSCN via two low gain antennae.
20. The communications system will be designed with a 4dB link margin.
21. The communications system will provide a Bit Error Rate of 10^{-6}.

22. The communications system will support an aggregate data rate of at least 16:1. The system shall be able to transmit data at 16 times the rate it is generated.
23. The command and data handling system will provide storage for at least 5 orbits worth of data.

STRUCTURES

24. The structures subsystem shall provide mounting provisions for all XTOS subsystems including the interface structure for the Minotaur launch vehicle. Provisions shall be made for suitable tie-down and lift attach points. These points shall not interfere with assembly checkout or integration.

9 MATE-CON Results

The ICE process used during the second half of this class showed some interesting advantages but also some drawbacks that the class had to deal with.

This process appeared to be human intensive, first because it needs a lot of people to run the whole process. We split the class into twelve different subsystems, and each person had to create an Excel sheet instantiation of his or her own subsystem. Each design study required the presence of every subsystem manager, plus a session director. Moreover, the ICE process cannot be automated because each subsystem has to be managed by a person able to detect any unreasonable value or result, and this person has to update and make the relevant changes for his subsystem. This also means that every subsystem manager needs to have the appropriate knowledge concerning his or her subsystem and should be trained in this field. In most cases, of course, the students did not meet these requirements, and were thus forced to simulate knowledge and experience through research.

The ICE process is also difficult to control, especially because a lot of different people are interacting. Each individual subsystem sheet is created and managed by a person, and each of these subsystem sheets have to interact throughout the ICE process. The control and the interaction of these twelve different subsystems was often difficult

Some difficulties also appeared during the convergence of the ICE process. Some design studies were sometimes slow and even difficult to converge, because each subsystem is, at its heart, a complex system of non-linear interactions; thus, the design sessions was an interaction between these complex non-linear systems. The interaction of two different subsystems acting on the same variable can lead to a divergence of the ICE process if the initial variables are not reasonable.

But the ICE process showed valuable advantages. First, it was very flexible. Once each of the subsystem sheets were ready and updated, any kind of change in the requirements could be handled very easily and very quickly. Any change could be made on the different variables or in the requirements; the results on the design would be studied easily.

This is a very fast learning process. Our class was composed of inexperienced people who never used this ICE process before, and within two months, our group learned and efficiently used this process.

There are still some remaining issues to be addressed. The first one is that the preliminary designs did not quite converge. The mass of the system was clearly following an exponential decay curve, but the class did not have time to follow this curve to a reasonably small change to call the design converged. These problems of convergence appeared as each subsystem was trying to find the most appropriate design for their own sheet.

The group also often needed to check through this process and make sure every variable and every result were reasonable. This sort of human control was needed frequently throughout the details of this process.

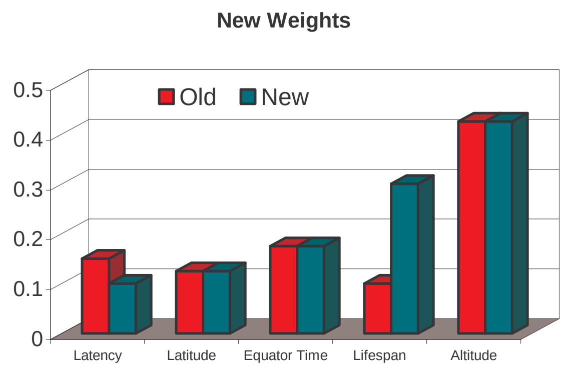
Some fine points or possible different solutions could not be studied through this process within the time constraints of a single semester. For example, the assumption of having cylindrical fuel tanks in the spacecraft was made from the beginning, but this general shape was not the optimal shape for fitting the most fuel inside of this particular spacecraft. In this case, a toroidal shape would have increased the amount of fuel carried. Another issue concerned the solar panels: the available outside area of the spacecraft appeared to be slightly insufficient to fit the total required solar panel area as body mounted cells.

One of the most promising attributes of MATE-CON is its flexibility, already mentioned. The original interviews for the MATE portion of the process are quite difficult and require iteration to be certain that the preferences have been accurately captured. In this particular case, we were unable to perform a validation interview until after the ICE process was well underway. Unfortunately, the validation interview showed us that the user preferences had not been captured accurately. However, this gave us a very realistic situation in which to test MATE-CON's flexibility: what happens to this process if the preferences change?

9.1 Changing User Preferences

MATE-CON is a process that is very adaptable to change, and in fact welcomes it. Feedback from the user is actually sought for to be sure that the designs are satisfying the user's true preferences. In the validation interview with Dr. Kevin Ray, some values from the first interview had in fact changed. Whether these changes were brought about by misconceptions in the first interview or actual changes in the user preferences does not matter. But being able to detect and incorporate these changes matters significantly. Hence the validation interview was a necessary and useful step in the MATE-CON process.

After reviewing the data from the first interview, Dr. Kevin Ray realized that he had not put enough emphasis on the spacecraft lifetime. The ability to capture many atmospheric cycles (such as day/night, monthly, yearly, and solar cycles) is actually quite important for a successful mission. This retrospection changed the weight factor of the data lifespan attribute from 0.1 (lowest) to 0.3 (second highest). In a similar manner, Kevin realized that there was very little importance on latency for a science mission. He reduced this weight factor from 0.15 (3rd highest) to 0.1 (lowest). Lastly, Kevin altered the shape of the data lifespan utility curve, which resulted in a somewhat linear relationship between utility and data lifespan. The utility of a 2-year mission was decreased from 0.35 to 0.3, and the utility of a 4-year mission was increased from 0.35 to 0.44. (Please see charts in Figure 9–1 for a clearer representation of these changes.)



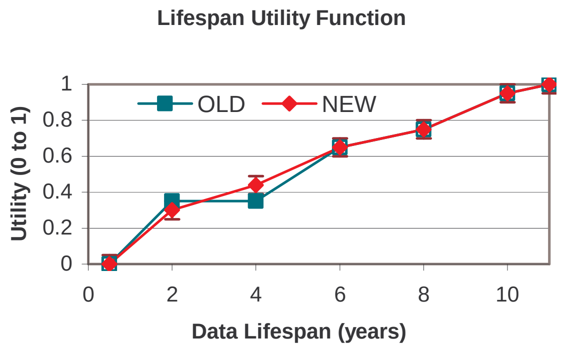


Figure 9–1: New weights and Lifespan utility function

These updated values from Kevin were estimates based on viewing the results from the MIST interview tool. Given more time, Kevin would be asked to run through the MIST software again to confirm that these are indeed his true preferences.

These drastic changes in user preferences, which might create huge setbacks for some design sessions, exploit the power of MATE-CON. These changes were immediately implemented into the utility software, and new utility functions were created within the day. The ICE design sessions followed suit and began searching for architectures based on the new utility information.

9.2 Final Results

Two of the final possible designs that converged are shown and briefly discussed below:

Table 9-1: MATE baslines and ICE designs acheived

	Mass (wet) [kg]	Pwr (avg) [W]	ΔV [m/s]	Apogee [km]	Perigee [km]	Life [yrs]	Utility (Orig)	Utility (Rev)	Life Cost [$M]
Original Base			1000	250	200	0.75	0.66		
ICE Result	449.6	164	1250	185	185	0.53	0.70	0.61	71.74
Revised Base			1000	350	350	9.8		0.70	
Current ICE	324.3	164	1000	300	300	2.20	0.59	0.55	75.01

The sections in Table 9-1 are alternating expected baseline and design achieved. In the first case, the “Original Base” shows the inputs from the architectural space from the MATE part of the class; the line below it, “ICE Result,” shows the design achieved in the ICE sessions. As can be seen, the first result exceeds the utility of the original expectations. The second result does not. This is due mainly to a lack of time; the semester finished before the class did. As is noted elsewhere in the paper (sections 5.4.4, 6), the first iteration on the customer’s preferences was completed rather late in the process. The resulting change in utility was incorporated into the design sessions almost instantaneously, with a re-run of the architectural study taking on the order of 45 minutes, and the design change taking on the order of several hours.

The class feels that these results validate the MATE-CON process in several ways: first and foremost, the class was able to quickly converge on a design very close to the baseline design desired. Secondly, because the class knew the user's preferences so well, it was able to leave the original tradespace by adding more fuel than originally calculated to achieve a higher utility. In mathematical terms, it was as if the class knew not only the user's preferences, but also the 'potential field' of preference: we knew what direction to drive the design to gain the most utility increase.

Due to the lack of time, the class was not able to achieve such a thorough understanding of the revised preferences. It is clear now that the way to manipulate the mission for higher utility is to increase the lifetime. Instead of this, the class developed different ideas independently for increasing the utility of the mission, such as overloading the fuel and using it to dynamically change the orbit of the spacecraft in response to the density and solar cycle launched into.

10 Lessons Learned / Insights:

10.1 MATE Process

The class really felt like their perceptions of Systems Engineering would never again be the same after they were exposed to the MATE process. The energy and creativity that was poured into this section of the class was a true testament to how much the members of 16.89 felt that they were involved in something truly groundbreaking. When the models finally came together and the first results were displayed (and after a huge sigh of relief was heard), a wave of ingenuity spread over the class – you could see minds churning and eyes scanning. No sooner than when the first utility plots were generated did powerful patterns and insights began to emerge from the data. A group of team members began to fervently attack the complex patterns and allowed the rest of the class to visualize the complex interactions that normally would not become apparent to traditional system engineers until one specific design had been chosen and almost fully developed. With the use of the MATE process, the team gained years of insight into this particular mission in a matter of a few short weeks.

With respect to the Matlab Code required to complete the MATE process, the class eventually decided that a completely new software architecture was necessary to complete the project in a timely manner. Learning from many of the basic equations and strategies that the previous models employed was not enough in this case to achieve the results that we desired within the time frame dictated by the class structure. In our case, this turned out to be a good thing – necessity *was* the mother of invention. After the process and project were a little better understood, several members of the class set out to create a MATE software architecture that could be run efficiently and intuitively. The result was the creation of a unique "satellite database" approach that was able to generate good data in small fractions of the time required by the C-TOS team.

While the unique approach to the MATE code worked elegantly in this particular case, we realized quickly that, just as the C-TOS architecture was not adequate for our needs, the X-TOS approach will not necessarily work for all future projects as well as it did for ours. After some discussion, the class generally felt that the MATE modules should be created from scratch for new projects (this does not mean that past projects should not be used as a learning guide!). If this is the case, however, there must be a critical mass of participants who feel comfortable programming in Matlab in order for the class to take advantage of the full power of the process. In addition, 16.89 members felt that some more time should have been dedicated up front to specific discussions (with *all* present) to the architecture of the code and the specific flows of information between modules. The class must also decide as a group the level of fidelity that they will be comfortable with, and an appropriate division of tasks so that no one person holds the key to the success or failure of the project.

The major problem encountered in the MATE section of the class was the delay in getting good information from the customer early on in the process. Although customer interaction can often be subject to various outside constraints, class members felt that it was imperative to begin to hear from the customer on the first day of the class in order to frame all future lectures and discussions.

Secondly, the class felt that the learning curve associated with a deep understanding of the MATE process was very shallow – we did not truly feel comfortable with the concepts and goals of MATE until we had actually finished the first half of the course. This brought with it a certain satisfaction of having worked through a difficult problem and creating a powerful tool along the way, but also a sense that our models could have been built more robustly and insightfully if more time or iterations were possible. Much of this discussion focused on the designation of the system Attributes, which represented the most fundamental departure from traditional systems engineering. As the class eventually found out, their choices at that critical point drove significant design changes later on in the process. These finding suggests that it may be valuable to have the class build their model in a couple of specifically-designed iteration loops of increasing detail.

The final major finding from the MATE process was a classic one – the cost of software is not necessarily proportional to the amount of blind faith that can be put into its output. In our case, several orbit vectors that were created using the Satellite Tool Kit were input into the initial Matlab models through the orbits database. After a more detailed examination, some major discrepancies were found in the data generated by the software (i.e. a 200 km ‘circular’ orbit ended up with over 10% variations in altitude over each orbit). The lesson learned from this error was that expert knowledge is still essential

10.2 ICE Process

The final version of our ICE sheets represented a strong, parametric satellite model. In general, the each member of the class was satisfied with the performance of their own worksheet, but was frustrated at the lack of time to fully explore the power of the tool they had developed.

The ICE process differed significantly from the MATE process in several ways. Obviously, having followed the MATE process, our concurrent engineering efforts began with a fairly deep understanding of the mission goals and key parameters. During this phase, the class devoted its time much more pointedly towards the technical details of the mission rather than on the process involved in attaining that data. This was mainly due to the relatively more mature format of the software architecture. Whereas the class really took the liberty to re-design the entire software architecture as it created its MATE models, the structure of the ICE spreadsheets remained unchanged from previous years work.

10.2.1 Technical Challenges

A number of lessons were learned by each of the subsystem chairs as the created, re-worked, piloted and engineered their way through the ICE process. The particular lessons learned have been documented within the appropriate subsystem sheet descriptions in the previous section.

10.2.2 Re-Use of Software Models

There was however a significant debate within the class as to the benefits of starting directly from the previous class’ ICE models (C-TOS in this case), or whether it would have been more productive to begin with blank spreadsheet templates that simply contained the macros for passing variables and possibly a list of potential variables to get started with.

A few of the subsystem chairs felt that they gained an invaluable head start by beginning with the C-TOS sheets. In particular, the Structures and Mission sheets only needed minor modifications. However, most of the other sheets required major changes, and the class felt that they spent a disproportionate amount of their time performing this task. Unfortunately, figuring out the logic of a highly complex Excel spreadsheet can take a significant amount of time. This is due in part to the format of the program (i.e. had to click inside formulas, check for cell numbers and hunt around to find what values those cell numbers represented), but also due to the fact that the equations and assumptions were not rigorously documented within the existing sheets. Due to the parametric nature of many of the sheets, many cells contained complicated IF statements and references to numerous other worksheets. When one change was made, such as deleting a “daughter” satellite from the legacy design, many links were broken resulting in the dreaded ”#REF.”

Finally, the class found that the sub-architectures of the sheets themselves were not consistent with each other. A primary example of this phenomenon was the inclusion of numerous “over-ride” buttons on the “input (general)” and “output (general)” in several of the subsystem worksheets, but not in others. As the concurrent design sessions unfolded, the presence or absence of these buttons became very important. If an error was encountered, and an estimate value could not be substituted, situations arose where the entire group was idle for a considerable amount of time while one small error was being fixed.

The uniqueness of each sheet could also have been attributed to a general feeling of uncertainty among class members regarding a rigorous approach to parametric modeling. Many 16.89’ers had different opinions concerning the robustness of the sheets. At a fundamental level, the debate centered on the inclusion of model limitations and whether inputs and outputs should be checked to see if they fit the acceptable ranges of the equations they were to be applied to. The level of detail contained within the subsystem sheets should be balanced carefully so all sheets are at the same fidelity.

After a prolonged discussion, the class concurred that the initial presence of the C-TOS sheet greatly accelerated the class’ movement down the ICE learning curve because participants were able to visualize how information would flow and understand the mechanics of passing variables and making detailed design trades. Once that learning occurs, however, the class felt that basing their individual sheets only off of standardized templates would be the best approach. These templates would contain all of the necessary macros, input and output pages, over-ride structures, and legacy design databases (such as detailed lists of launch vehicles and propulsion components).

10.2.3 Process Flow

After the ICE sheets were mature enough to begin running design trades, class members began to see the power of the process. Not only did we find that designs could be quickly and accurately generated, but we had a method of evaluating them against each other and mapping out our trade studies (via our MATE chair). During the integrated design sessions, the class worked extremely well together to address inconsistencies and reason through unexpected results. The projected maps of system mass breakdowns and the basic configuration layout were extremely helpful for all involved.

Some general insights into the ICE process were a result of numerous design sessions:

- The preparation of each team member is crucial to a smooth running session – the loss of one sheet created unacceptable delays. Likewise, every team member needs to be present for the process to flow smoothly.
- Complex interactions between subsystem sheets were not detected by individual debugging efforts. Subgroups needed time to work together to understand how their sheets were interdependent.
- The co-location of all participants within one room was seen as extremely important. Several team members had worked on the C-TOS project, which included team members who worked over the Internet. They all concurred that face-to-face interactions were far superior in obtaining accurate and fast-paced decisions during the early stages of the process.
- The sequencing of the send/receive process could lead to either long delays or premature assumptions of convergence. The class experimented with several ‘protocols’ regarding the order that certain subsystem sheets synchronized with the ICE server. If a general announcement was made to “send and receive” it usually turned out that some of the sheets would be finished before others and thus their last calculations were not based on the latest information from other sheets. Another method was to have the ICE “conductor” signal individually to each sheet when to send or receive information. This generally led to convergence around a particular design point within four or five iterations, however this could take from 5 to 15 minutes. The class recommended writing some sort of macro to automatically synchronize the data.

11 Conclusions

11.1 Process Summary

The overall process followed throughout the semester can be summarized as follows. In the first part of the process (MATE: Multi-Attribute Tradespace Exploration), the mission is defined and a list of attributes reflecting the user's preferences is created. Interviews are then conducted with the user to discover the relative importance of each attribute to the user. The results are combined into a mathematical utility function that gives the value of user utility as a function of each attribute.

From that point, a list of design variables is produced. Each combination of design variables represents a design vector that will ultimately produce a system architecture. A Quality Function Deployment (QFD) exercise is used to map the design vector to the system attributes and to eliminate extraneous variables that might make the design vector unmanageable. The design space is then defined by determining appropriate ranges for the design variables based on physical and system constraints.

The computational model of the system partitions the problem into modules that calculate attributes based on design vector inputs. Teams are created to develop each module or set of modules, and an integration team is created to keep track of inputs and outputs, ensure communication, and assemble all the modules into a full working model of the system. The software model compiles a database of thousands of satellites for further cost and utility analysis. Satellites are then combined using a set of mission scenarios to define thousands of architectures.

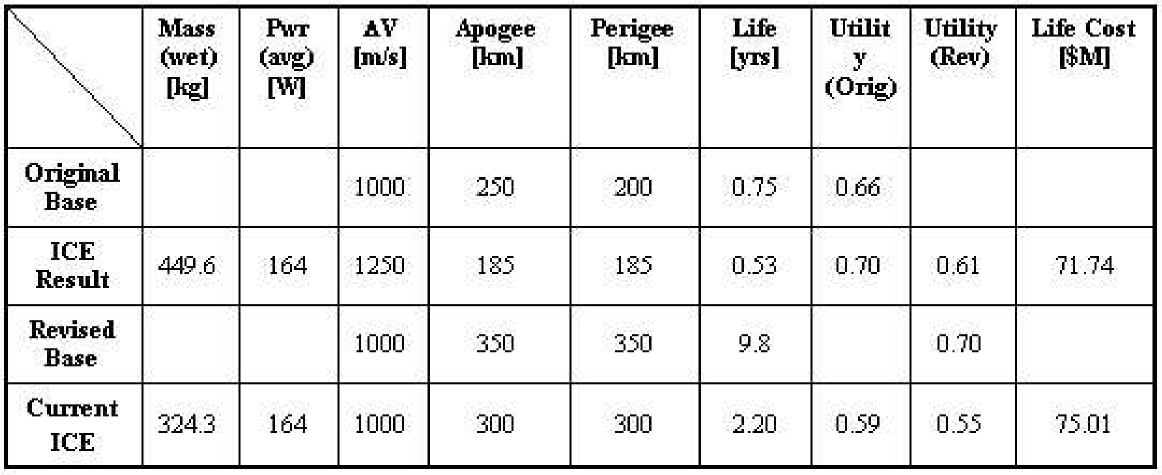
The cost and utility models are then used to evaluate all possible meaningful architectures. The architectures are first compared in terms of utility and lifetime cost, and those architectures that have both a high utility and an acceptable cost are examined with a greater level of detail. One particular pareto optimal system architecture is finally selected and propagated to the second part of the process – the concurrent design phase (CON or ICE, for Integrated Concurrent Engineering).

The tradespace is analyzed in greater detail in the second phase of the process. ICEMaker, a software tool linking independent Microsoft Excel Workbooks through a common server, is used to facilitate the interaction among the different subsystems. All team members operate in the same environment, both physically and technically. Each team member is responsible for one subsystem and may suggest key trades that will affect his/her subsystems and the overall spacecraft performance. Real-time feedback into the effect of each trade on the other subsystems and on the overall utility and cost of the mission is easily disseminated. MATE-CON is indeed a single, integrated process. As designs are generated during an ICE session, they are directly mapped back to the MATE phase of the process and their utility is assessed. The results are then used to choose the appropriate trades to examine in greater depth. After a great deal of iteration, a converged design is obtained.

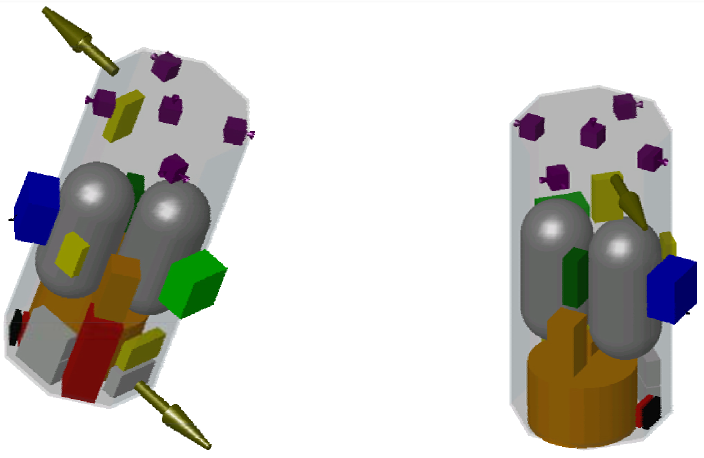
11.2 Results Summary

At the end of the MATE-CON process an architecture and a point design were obtained. Because the utility function was modified after the validation interview, as described in Chapter 9.1, the results are given below for each of the two sets of user preferences.

Figure 11–1 shows the expected baseline and the design achieved after concurrent engineering for both the original utility and the modified utility. In the first case, the "Original Base" shows the inputs from the architectural space from the MATE part of the class; the line below it, "ICE Result," shows the design achieved in the ICE sessions. In the second case, "Revised Base" shows the inputs from the MATE section, and "Current ICE" shows the design resulting from the concurrent process at the time the class ended. In both cases the spacecraft is to be launched on a Minotaur launch vehicle.



	Mass (wet) [kg]	Pwr (avg) [W]	ΔV [m/s]	Apogee [km]	Perigee [km]	Life [yrs]	Utilit y (Orig)	Utility (Rev)	Life Cost [$M]
Original Base			1000	250	200	0.75	0.66		
ICE Result	449.6	164	1250	185	185	0.53	0.70	0.61	71.74
Revised Base			1000	350	350	9.8		0.70	
Current ICE	324.3	164	1000	300	300	2.20	0.59	0.55	75.01



(a) (b)

Figure 11–1: Satellite configuration for design using original (a) and modified (b) utilities

11.3 Accomplishments

Throughout the course of this project, the class noted some important accomplishments. In a relatively short period of time, user preferences were captured and thousands of system architectures were evaluated. A host of architectures that met the user's needs were identified and a point architecture was selected. The efficiency and usefulness of the MATE-CON process were demonstrated by completing the entire architecture selection and preliminary design process in just three months. The flexibility of the process was also demonstrated by the rapidity at which changes in user preferences were captured and fed back into the design process to yield a new result.

In addition to the results of the architecture study, the robust and reusable Matlab code and ICE sheets that were developed will prove to be useful tools for future Space Systems Engineering classes.

11.4 Lessons Learned

Many lessons were learned throughout the process of completing this project. Many of the lessons learned were covered in previous sections. In this section, the major insights or lessons learned from the class are highlighted.

Perhaps the most important lesson to be passed on is that consistent and clear communication within the team, faculty, and users/customers is indispensable to the success of the project. In particular, it is important to validate the results and assumptions made with the users/customers on a regular basis. The shift in the utility curve during the second part of the class could not have been discovered had the MATE results not been validated. This leads to another important lesson – early contact with the users/customers is absolutely essential. The delay in getting good information from the customer early on in the process was indeed one of the major problems the class encountered in the MATE section. It is also very important to facilitate communication within the team itself. The co-location of all participants within one room proved to be extremely valuable, but the class felt more time should be spent to make sure all team members share the same mental model of the process, the software architecture and the information flow.

One of the primary issues that arose with the structure of the course centered on re-usability of the MATE and ICE models from previous designs. The availability of past projects greatly facilitated the class' learning because participants were able to visualize how information would flow and have an overall view of the process. Understanding the mechanics and the models used in the individual modules and subsystem sheets was, however, very time-consuming. Model reuse seemed to be inefficient in some cases, or even wrong when the underlying assumptions were different. Reuse can be dangerous without critical judgment. Complex interactions between subsystem sheets or modules were also difficult to manage, and should be examined more carefully in the future.

The need for flexibility or agility was another primary lesson learned from the class. The flexibility of the process was demonstrated when the class was able to adapt very quickly to the change in user preferences. After the validation interview with the customer, a new utility function was derived. Due to the parametric nature of the ICE models, the design was modified within one session to meet the new preferences! In addition to the process, the participants themselves had to be very flexible. For instance, as the utility team was defining the list attributes in the early stages of the class, several changes had to be made and the team members were able to adapt quickly and modify their models accordingly. The team members had also to be agile. The configuration chair in the ICE process for instance needed to visually examine the satellite configuration at each iteration, and any delay in this particular step would have delayed the design process.

The final learning from the class concerned the level of fidelity needed. The models developed were not required to have a very high level of fidelity since the purpose was to develop a preliminary design rather than a detailed one. A sufficient level of fidelity had to be ensured, nonetheless, for the results to be meaningful. Some sub-teams had to modify their models to provide the overall level of fidelity needed, since the simulation is only as accurate as its weakest link. This is a basic engineering problem and the class was able to deal with it thanks to the flexibility of the process.

11.5 Recommendations

Although the process was found to be very efficient, useful, and flexible, the class identified several future opportunities to improve the MATE-CON process.

On the technical level, the class felt improvements could be made concerning the data transfer sequencing in the second phase of the process. N2 diagrams and Input/Output documents facilitated system integration in the MATE section; similar tools are needed to sequence the "send/receive" actions and thus improve the convergence rate for ICE. Improvements can also be made concerning the debugging of sheet interactions in the ICE section. Several methods were used to verify and validate the individual sheet and to test the model as a whole, but debugging was very difficult at a system level and a more systematic approach is needed for the future.

As mentioned in the previous section, the effect of reuse on the process efficiency and quality should be assessed. It was not clear to the class whether using the BTOS and CTOS models as a baseline was an advantage or not. This issue should be examined more closely. Also, intra-team communication should be further facilitated to increase the learning curve slope. In particular, the class felt it would have been useful to dedicate some sessions exclusively to this purpose, ensuring that everyone in the team understood the process, the software architecture, and the information flow.

The class would have also liked to be able to quantify some of the uncertainty surrounding the results obtained. Sources of uncertainty are indeed numerous, from technical and political uncertainty to simply the uncertainty inherent to the models used and the simplifications made. Being able to quantify some of this uncertainty would definitely increase confidence in the results.

Finally, the class believes that incorporating some of the recent work done on risk assessment would add value to the process. In particular, the use of portfolio theory to assess the value of dynamically adjusting the mission is very appealing and could allow for original mission concepts in the future.