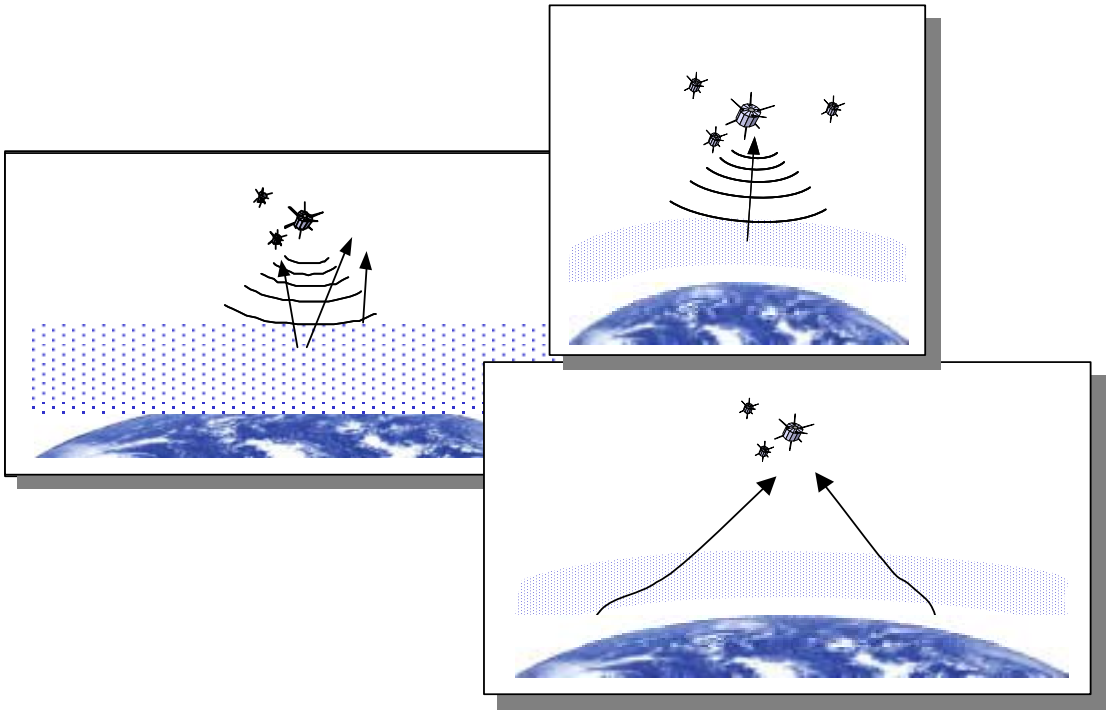
16.89J / ESD.352J Space Systems Engineering
Spring 2007

B-TOS

Terrestrial Observer Swarm



Massachusetts Institute of Technology
16.89 – Space Systems Engineering
Spring 2001

Final Report

B-TOS Architecture Study

Second Iteration of the Terrestrial Observer Swarm Architecture

Document Number	1.0
Document Manager	Michelle McVey
Authors	Nathan Diller, Qi Dong, Carole Joppin, Sandra Jo Kassin-Deardorff, Scott Kimbrel, Dan Kirk, Michelle McVey, Brian Peck, Adam Ross, Brandon Wood
Under the Supervision of	Prof. Dan Hastings, Prof. Hugh McManus, Prof. Joyce Warmkessel

Course 16.89 - Space Systems Engineering
Department of Aeronautics and Astronautics
Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139

Acknowledgments

Professor Joyce Warmkessel, Professor Hugh McManus, and Professor Dan Hastings for instructing the class and providing guidance throughout the course.

Col. Keesee, Professor Sheila Widnal, Professor David Miller, Dr. Ray Sedwick, and Dr. Joel Sercel for their lectures on Space Systems and further assistance in their areas of expertise.

Fred Donovan for his computer assistance and acquiring licenses for Satellite Tool Kit.

Dr. Bill Borer and Kevin Ray for providing the team with an "aggregate customer" and all of their time and support.

Mr. Pete Hendrickson and Ms. Nicki Nelson for providing feedback at our architecture design review.

Mr. Myles Walton for his contributions to our code development.

Dr. Bill Kaliardos for his contributions to our code development as well as our process documentation.

Contents

List of Figures

List of Tables

Acronym List

A	Accuracy
AFRL	Air Force Research Laboratory
AOA	Angle of Arrival
A-TOS	First study for the design of a Terrestrial Observer Swarm
BER	Bit Error Rate
BOL	Beginning Of Life
BPS	Bit Per Second
B-TOS	Second study for the design of a Terrestrial Observer Swarm
C.D.H	Command Data Handling
CAD	Computer Aided Design
CER	Cost Estimating Relationship
C-TOS	Third study for the design of a Terrestrial Observer Swarm
D	Daughtership
DSM	Design Structure Matrix
DSS	Distributed Satellite Systems
EDP	Electron Density Profile
EOL	End of Life
FOV	Field Of View
GINA	Generalized Information Network Analysis
GPS	Global Positioning System
GUI	Graphic User Interface
HF/LVHF	High Frequency/HR
I/O	Inputs/Outputs
ICE	Integrated Concurrent Engineering
IGC	Instantaneous Global Coverage
INT	Integer value
IOC	Initial Operating Capability
ISS	International Space Station
L	Latency
LEP	Lottery Equivalent Probability
LV	Launch Vehicle
M	Mothership
MAU	Multi Attribute Utility
MAUA	Multi Attribute Utility Analysis
Mbs	Mega Bits per Second
MC	Mission Completeness

MOL	Middle Of Life
MTTF	Mean Time To Failure
QFD	Quality Function Deployment
RAAN	Right Ascension of the Ascending Node
RT	Revisit Time
SMAD	Space Mission Analysis and Design
SR	Spatial Resolution
SSPARC	Space Policy and Architecture Research
STK	Satellite Tool Kit
STS	Space Transportation System
TDRSS	Tracking and Data Relay Satellite System
TEC	Total Electron Content
Tx/Rx	Transmit, sound/Receive capacity
UML	Universal Modeling Language
UV	Ultra Violet

1 Executive Summary

The B-TOS project, using the evolving SSPARC method, may change the way in which conceptual design of space-based systems takes place in the future. This method allows for rapid comparison of thousands of architectures, providing the ability to make better-informed decisions, and resulting in optimal solutions for mission problem statements. The process was completed and results were obtained by the 16.89-Space Systems Engineering class during the spring semester of 2001. The class addressed the design of a swarm-based space system, B-TOS (B-Terrestrial Observer Swarm), to provide data for evaluation and short-term forecasting of space weather. The primary stakeholders and participants of the project are 16.89 students, faculty and staff, and the Air Force Research Laboratory (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 ionospheric behavior, thus allowing improved global communications for tactical scenarios. Secondly, from a pedagogical standpoint (student and faculty), the class serves as a testing ground for the evaluation of a new and innovative design process, 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 B-TOS mission, as well as identification of top-level system requirements based on the stakeholder constraints and user wants and needs. 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.

In order to fulfill AFRL needs for an ionospheric forecasting model, the B-TOS satellite system is required to perform three primary missions:

1) Measurement of the topside electron density profile (EDP)
2) Measurement of the angle of arrival (AOA) of signals from ground-based beacons
3) Measurement of localized ionospheric turbulence

To perform these missions, the system is required to use a swarm configuration, maintain a minimum altitude for topside sounding (to operate above the F2 peak in the ionosphere), operate at a frozen orbital inclination of 63.4º, and use TDRSS for communication with the ground. Additionally, each of the satellites within the swarm must be capable of housing a black-box payload for an additional non-specified customer mission. An evolved GINA/SSPARC design process is utilized to develop a large set of space system architectures that complete mission objectives, while calculating customer utility, or relative value of each, as weighed against cost. This design process eliminates missed solution options that result from focusing on a point design. Instead, it gives to the primary user a host of choices that can be juxtaposed against each other based on their relative value. The system model has the capability to predict customer utility by varying orbital geometries, number of swarms and size, swarm density, as well as the functionality of individual satellites. The level of detail was chosen based on the resources of this class project and the necessity to accurately distinguish relevant differences between competing architectures.

Upon completion of the design process, a series of architectures were determined to be viable to complete the mission and satisfy user needs. One of the most promising architectures considered is a 10-satellite system for a total cost of $263 million over a 5-year lifecycle. The system consists of two types of satellites: 9 daughtership satellites with limited capability, and 1 mothership with enhanced communication and payload capabilities. A requirements summary for this configuration is presented, as well as a sensitivity study to the model constraints and assumptions. Finally, this report contains lessons learned from the entire class process, as well as a documented version of the master program used to study architecture trades.

2 Introduction

The purpose of this document is to describe and summarize the process completed and results obtained by the 16.89 class during the spring semester of 2001. The class addressed the design of a swarm-based space system, B-TOS, to provide data for evaluation and short-term forecasting of space weather. The primary stakeholders and participants of the project are: 16.89 Students, faculty and staff, and AFRL. Furthermore, the Space Policy and Architecture Research Center (SSPARC) is also interested in seeing the implementation of the Multi-Attribute Utility Analysis (MAUA) for a real space system.

2.1 Motivation

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 ionospheric behavior, thus allowing improved global communications for tactical scenarios. Secondly, from a pedagogical standpoint (student and faculty), the class serves as a testing ground for the evaluation of a new and innovative design process while teaching and learning the fundamentals of space system design.

2.2 Objectives

The objectives of 16.89 are for the students to develop and justify a recommended space system architecture and top-level system requirements, based on stakeholder constraints and user needs and wants. Functional, design, and operational requirements are established for both the ground and space segments, as well as a preliminary design for the space component.

2.2.1 Mission Statement Development

The mission statement for the B-TOS project was developed through class and faculty iteration. The key features of the mission statement are to articulate:

- ***What*** the project is about?
- ***Why*** should the project be undertaken?
- ***How*** the project will be done?

The B-TOS mission statement is:

> ***Design a conceptual swarm-based space system to characterize the ionosphere. Building upon lessons learned from A-TOS, develop deliverables, by May 16, 2001, with the prospect for further application. Learn about engineering design process and space systems.***

The deliverable mentioned above refers to the B-TOS reusable code, final report, and requirements document.

2.2.2 Assessment Methods

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.

To assess the success of this design process, four formal reviews were completed, with this report documenting this process. The table below summarizes the key milestones that are used to assess the class progress.

Table 2-1 B-TOS Milestone Dates

Review Name	Date	Purpose
Progress Review	3/5/01	Review to present the approach that is used to conduct the B-TOS architecture evaluation. The utility function and initial input vector are specified, as well as descriptions of the B-TOS modules.
Midterm Process Review	3/21/01	The purpose of this review is to assess the class understanding of the architecting process and background material that has been presented to the class to date.
Architecture Review	4/9/01 and 4/18/01	This review presents the results of the architecture evaluations. The review establishes the initial architecture that is chosen to the spacecraft design.
Final Review	5/16/01	This is the final review of the culmination of the class project and presents a summary of this document, with emphasis on the final B-TOS architecture and selected design.

Furthermore, it was stated that student's completing 16.89 will be able to develop and justify recommending system architectures and top-level system requirements based on stakeholder constraints and user wants/needs, and be able to state functional and design and operational requirements for the space segment.

2.2.3 Class Value Proposition

At the outset of the class, the following two questions were posed to the class by the faculty to garner an understanding of what the class is most interested in:

1. What do you want from the class?
2. What do you expect to contribute to class
    a. Level of effort
    b. Special interests
    c. Special expertise

As expected, these interests were dynamic. Over the course of the semester the faculty provided the class several opportunities to re-define the direction in order to meet expectations.

2.3 Approach

Our basic approach was to learn the scientific purpose of the space system and develop a framework for the development of a system to meet that purpose. Several constraints were placed upon the system. In order to make this a problem that could be adequately approached in the allotted time, considerations regarding the priorities of the class were defined. In general the class approached this problem using the Space System, Policy, and Architecture Research

Center's (SSPARC) evolved Generalized Information Network Analysis (GINA) method. The GINA method for Distributed Satellite Systems (DSS) system-level engineering was developed by MIT's Space Systems Laboratory, and enables the creation and comparison of many different design architectures for a given mission. The GINA method formulates satellite systems as information transfer networks. The SSPARC method evolves the GINA method by using customer value as the output metric, rather than information-based metrics that may have little or no meaning to the customer.

2.3.1 B-TOS Mission Overview and Scope

The general purpose of the B-TOS mission is to characterize the structure of the ionosphere using topside sounding. The topside sounding is conducted from a space-based platform. The development of that optimal platform is the focus of this report. Once the data is collected, it will be sent to AFRL's modeling systems to map the ionosphere for a variety of science and military users. The three primary missions are completed by the space system:

1. *Measurement of electron density profile (EDP)*
2. *Beacon Angle of Arrival (AOA)*
3. *Measurement of ionospheric turbulence*

Additionally, each of the satellites within the swarm must be capable of housing a special black box payload.

The general purpose of the B-TOS mission is to characterize the structure of the ionosphere using topside sounding. The topside sounding is conducted from a space-based platform. The development of that optimal platform is the focus of this report. Once the data is collected, it will be sent to AFRL's modeling systems to map the ionosphere for a variety of science and military users.

Motivation for Ionospheric Forecasting:

The ionosphere is the region of the Earth's atmosphere in which solar energy causes photoionization. This causes growth in the ionosphere during the day but because of low gas densities, recombination of ions and electrons proceeds slowly at night. It has a lower altitude limit of approximately 50-70 km, a peak near 300 km altitude and no distinct upper limit, as can be seen in Figure 2-1.

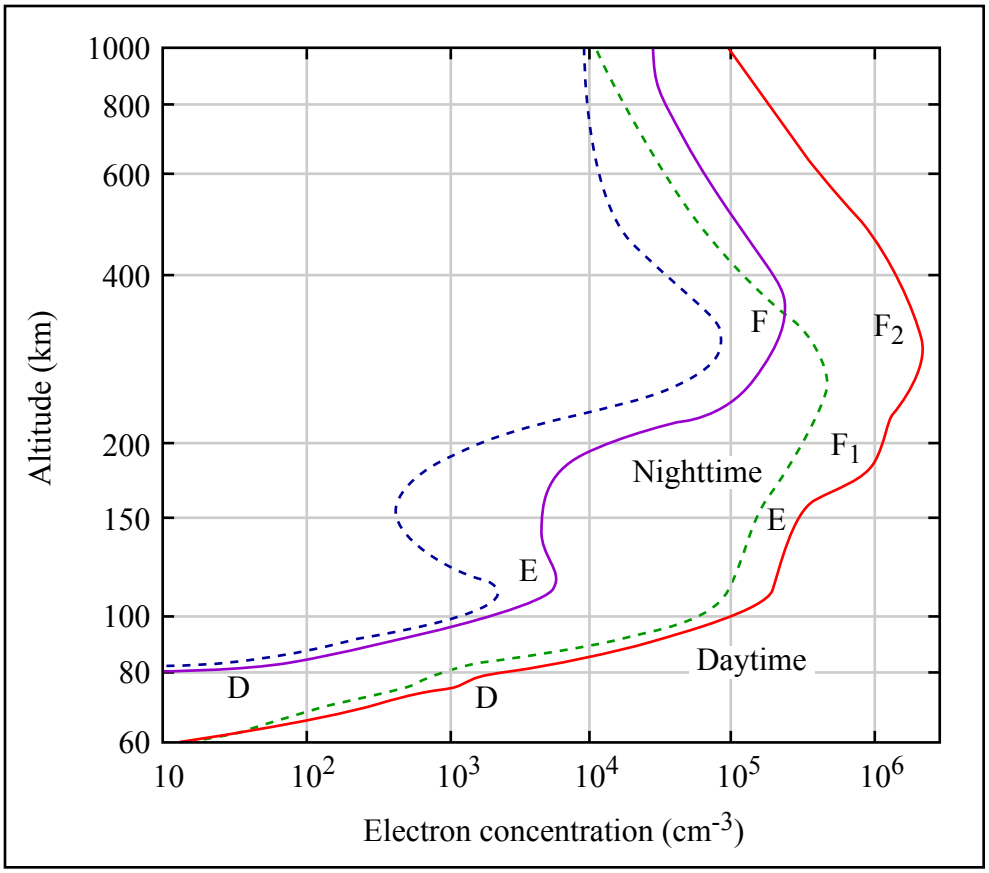


Figure by MITOpenCourseWare.

Figure 2-1 Day and Night Electron Concentrations[1]

The diurnal variation of the ionosphere directly impacts the propagation of radio waves through the ionosphere. The climatology of the ionosphere is well known, but the daily ionosphere weather, and therefore the effects on radio communication, evades prediction. Depending on frequency, the impacts can range from phase and amplitude variations to significant refraction and scintillation. These effects can cause loss of GPS lock, satellite communication outages, ground to space radar interference and errors, and HR radio outages. The turbulence in the ionosphere is often concentrated around the magnetic equator, so the radio propagation errors are most common around the equator.

Ionospheric Measurement Techniques

There are a number of techniques available to measure relevant parameters of the ionosphere. Ground-based ionosondes, which measure F2 altitudes from the surface, are commonly used today but they measure the electron density profile only up to the region of peak density (the F2 region on Figure 2-1). A number of space-based techniques are available as depicted in Figure 2-2.

[1] T. Tascione, *Introduction to the Space Environment*, 1994.

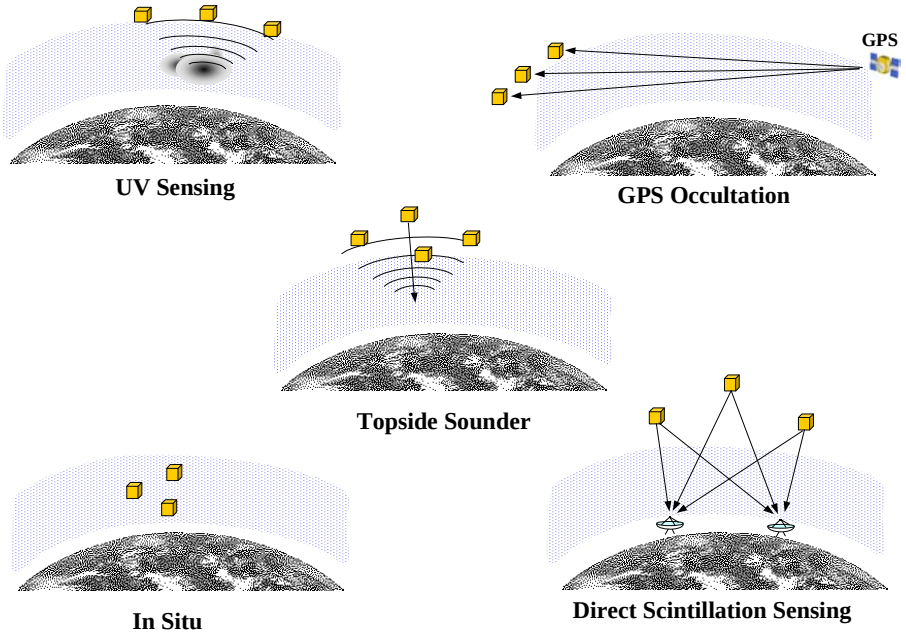


Figure 2-2 Ionosphere Measurement Techniques

The first potential technique involves detection of the ultraviolet radiation emitted by ionospheric disturbances. Viewing the UV radiation on the night-side is much less complicated than on the day-side and experts debate whether useable dayside measurements can be made. GPS occultation involves the measurement of dual GPS signals to provide data to calculate a horizontal measurement of the total electron content (TEC) between the receiving satellite and rising and setting GPS satellites. This orientation is significant because a horizontal slice of the ionosphere is more homogeneous than a vertical slice. A variety of instruments can gather ion and neutral velocity data while in situ. Combining this data with electric field and plasma density, also done in situ, has the potential to provide sufficient data for forecasting models. Ground based receivers are also used to measure radio wave scintillation and therefore ionosphere variability. The final measurement technique, topside sounding as represented in the center of Figure 2-2, relies on spacecraft orbiting above the ionosphere. It acts similar to an ionosonde, but collects electron density profile data, as can be implied, from the topside of the ionosphere. Since ionosphere variability often results in disturbances rising above the peak density region, a topside sounder has the potential to collect very valuable forecasting data.

B-TOS Payload Instruments

The payload on the B-TOS satellites has a combination of the aforementioned instrument types. The primary payload is a topside sounder that measures the electron density profile (EDP) between the satellites altitude and the peak density region by cycling through a series of frequencies and timing the reflection from the ionosphere. This instrument is also capable of collecting total electron content data in the nadir direction by measuring radio wave reflection off the surface of the earth. The second instrument in the B-TOS payload measures signals

propagated through the ionosphere from ground-based beacons. The ionosphere's refractive index can be calculated by comparing the true angle between nadir and the beacon's location with the measured value. The third ionosphere-measuring technique, used in conjunction with other satellites in the B-TOS swarm, is able to measure off-nadir turbulence in the ionosphere. Knowledge about the small-scale structure is valuable for scintillation prediction models.

Additionally, each of the satellites within the swarm must be capable of housing a special black box payload. Designated payload "B," the design team was given no information about this payload, other than what is necessary for sufficient integration into the rest of the satellite.

2.3.2 B-TOS Priority Matrix

The purpose of the B-TOS priority matrix is to focus the class on four key issues associated with the project: scope, schedule, fidelity (rigor) and resources and to balance these against each other to determine what is most important. The B-TOS priority matrix is shown below:

Table 2-2 Class Priority Matrix

	High	Medium	Low
Scope		X	
Schedule	X		
Fidelity			X
Resources		X	

The class decided that the most important of these was to keep the schedule on track, while considering a good portion of the scope of the B-TOS project. Resources, including people, unique knowledge, tools and training were determined to be at the medium level, while it was decided that the fidelity of the code could be somewhat lower, but still maintain the amount necessary to perform realistic and valuable systems trades of the architectures.

2.3.3 Notional Flow

To design such a system, an innovative design process is 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 other solution options by focusing on a point design, but rather gives to the primary user a host of choices that can be juxtaposed against each other based on their relative value.

Figure 2-3 B-TOS Notional Flow Diagram

Figure 2-3 shows the notional flow followed in B-TOS. Below is a basic description of each of the different facets of this process.

- **Design Space / Design Vector (Chapter 4):** Provides the available (variables) trades that were varied to find the optimal architectures. In B-TOS these variables included **Orbit level**-altitude, number of planes, and number of swarms per plane; **Swarm level**-number of satellites per swarm and radius of swarm; **spacecraft**-payload transmit, payload receive, on-board processing, long-range communication (TDRSS Link), intra-swarm link
- **Constants Space / Constants Vector (Chapter 5 & 6):** These are the different constants were used in the modules. Some of these constants are well-known but others need further research with the model having a variable sensitivity to each.
- **Model / Simulation (Chapter 5 & Appendix E):** Takes a possible architecture defined by the design vector, using computer code measures the attributes of that particular configuration.
- **Attributes (Chapter 3):** Six performance measurements in which the customer is interested. These attributes include instantaneous global coverage, latency, revisit time, spatial resolution, accuracy, and mission completeness.
- **Utility Function (Chapters 3 & 5):** Defines a single utility based upon the customer's preference for each of the attributes.
- **Cost & Utility:** The final outputs of the model, which are typically plotted with one another to create a focused tradespace.

2.3.4 Results

Upon completion of the design process, a series of architectures were determined to be viable to complete the mission and satisfy user needs. MAUA was successfully implemented providing the customer with a focused tradespace of architectures to meet the desired architecture attributes. Ultimately, a conceptual swarm-based space system to characterize the ionosphere was developed, by building upon lessons learned from A-TOS. Presentations, the Matlab code, and this document, which will all be complete by May 16, 2001, can be used for further application. The entire process facilitated student learning in the fields of engineering design process and space systems.

3 Multi-Attribute Utility Analysis

3.1 Background and Theory

A fundamental problem inherited from A-TOS was the need to determine the "value" of an architecture to the customer. The "value" and cost of each architecture were to be the primary outputs of the A-TOS tool. In A-TOS this was captured through the "value" function that assigned accumulated points each time the architecture performed "valuable" tasks in the course of a simulation. Two missions were identified for A-TOS: a high latitude mission, and a low latitude mission. Each architecture would get a score for each mission. The score for the low latitude mission ranged from 1-8. The score for the high latitude mission ranged from 1-200, though there was no hard upper bound. Results of the simulations were plotted in three dimensions: high latitude value, low latitude value, and cost. (Note: The word "value" is used here, when in fact the word "utility" was used in A-TOS. For reasons of clarity, the word "utility" will only be used to refer to the utility analysis discussed below.)

Several problems plagued the A-TOS value capture method. First, the scales of worst and best values for the value of an architecture were arbitrary. The values could be normalized, however due to the lack of a hard upper bound on the high latitude utility, the normalization would not be strictly correct. Additionally, there was at first no ability to compare the two separate values. Does a 0.8 high latitude value correspond to a 0.8 low latitude value? Further interviewing with the customer revealed that he valued the low latitude mission "about twice" that of the high latitude mission. This information led to an iso-value curve on a high latitude value versus low latitude value plot of 2 to 1.

$$V(X) = g(X_1, X_2, ..., X_n) \qquad \text{high latitude value}$$

$$V(Y) = h(Y_1, Y_2, ..., Y_m) \qquad \text{low latitude value}$$

Additionally, a total architecture value variable was defined as a weighted sum of the two separate mission values.

$$V(X,Y) = a_X V(X) + a_Y V(Y)$$

Total value = high latitude value + 2*low latitude value

The problem with linear weighting is that it does not account for tradeoffs in value to the customer. *Complementary goods* will both result in higher value if both are present together. *Independent goods* will not result in additional value based on the presence of another good. *Substitute goods* will result in lower value if both are present, with it preferred to having one or the other present. These effects would be present in a multi-linear value function.

$$V(X,Y) = a_X V(X) + a_Y V(Y) + a_{XY} V(X)V(Y)$$

In this case, if $a_{XY} > 0$, X and Y are complements; if $a_{XY} < 0$, X and Y are substitutes; if $a_{XY} = 0$, there is no interaction of preference between X and Y. However, this form was not used in A-TOS. It was assumed that there was no interaction of preference. The lack of a rigorous value-capture and representation process in A-TOS resulted in an unsettling weakness of the results. (At least in an academic sense.) A more formal and generalized approach was needed for measuring utility in B-TOS.

3.1.1 Motivation

Two members of 16.89 had taken Dynamic Strategic Planning in the Fall at MIT and were exposed to Multi-Attribute Utility Analysis (MAUA). This theory is a good replacement for the "value" function used in A-TOS. It provides for a systematic technique for assessing customer "value", in the form of preferences for attributes. Additionally, it captures risk preferences for the customer. It also has a mathematical representation that better captures the complex trade-offs and interactions among the various attributes. In particular, the strength of multi-attribute utility analysis lies in its ability to capture a decision-maker's preferences for simultaneous multiple objectives.

A key difference between a "value" and a "utility" is that the former is an ordinal scale and the latter a cardinal one. In particular, the utility scale is an *ordered metric scale*. As such, the utility scale does not have an "absolute" zero, only a relative one. One consequence of this property is that no information is lost up to a positive linear transformation (defined below). It also means that the ratio of two numbers on this scale has no meaning, just as a temperature of 100°C is not four times as hot as a temperature of 25°C. (The Celsius scale is an example of an ordered metric scale[2].)

Another difference is that "utility" is defined in terms of uncertainty and thus ties in a person's preferences under uncertainty, revealing risk preference for an attribute. It is this property along with other axioms that result in a useful tool: a person will seek to maximize expected utility (unlike value, which does not take into account uncertainty)[3]. This definition gives utility values meaning relative to one another since they consider both weighting due to the attribute and to continuous uncertainty. In summary, the value function captures ranking preference, whereas the utility function captures relative preference.

Before continuing, the term "attribute" must be defined. An attribute is some metric of the system. The power of MAUA is that this attribute can be a concrete or fuzzy concept. It can have natural or artificial units. All that matters is that the decision-maker being assessed has a preference for different levels of that attribute in a well-defined context. This powerfully extends the A-TOS value function in that it translates customer-perceived metrics into value under uncertainty, or utility. For B-TOS, the utility team felt that the utility function would serve well as a transformation from metric-space into customer value-space.

After iteration with the customer, the finalized B-TOS attributes were Spatial Resolution, Revisit Time, Latency, Accuracy, Instantaneous Global Coverage, and Mission Completeness. (For more information about the evolution and definition of the attributes, see below.) The first five attributes had natural units (square degrees, minutes, minutes, degrees, and % of globe between +/- inclination). The last attribute had artificial units (0-3) defined in concrete, customer-perceived terms.

The process for using utility analysis includes the following steps:

1. Defining the attributes
2. Constructing utility questionnaire

[2] Richard de Neufville, Applied Systems Analysis: Engineering Planning and Technology Management, McGraw-Hill Publishing Co., New York, NY (1990). (See chapter 18 for a discussion regarding value and utility functions.)
[3] Ralph L. Keeney and Howard Raiffa, Decisions with Multiple Objectives: Preferences and Value Tradeoffs, John Wiley & Sons, New York, NY (1976). (See chapter 4 for a discussion of single attribute utility theory.)

3. Conducting initial utility interview
4. Conducting validation interview
5. Constructing utility function

These steps are discussed in more detail in the following sections. The remainder of this section will address the theoretical and mathematical underpinnings of MAUA.

3.1.2 Theory

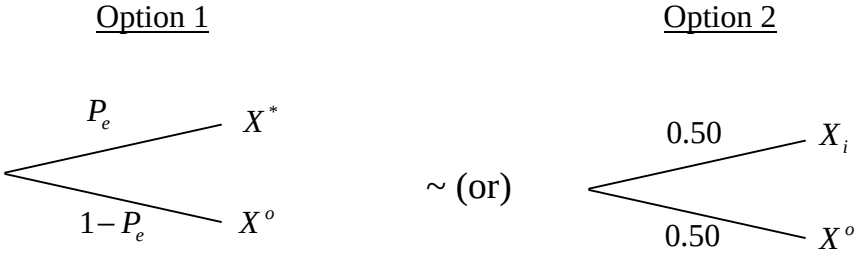
As mentioned previously, a utility function, $U(X)$, is defined over a range of an attribute X and has an output ranging from 0 to 1. Or more formally,

$$0 \le U(X) \le 1, \qquad X^o \le X \le X^* \text{ or } X^* \le X \le X^o$$

$$U(X^o) \equiv 0 \qquad U(X^*) \equiv 1$$

X^o is the worst case value of the attribute X.
X^* is the best case value of the attribute X.

Single attribute utility theory describes the method for assessing $U(X)$ for a single attribute. (von Neumann-Morgenstern (1947) brought this theory into modern thought[4].) Applied Systems Analysis refines this method in the light of experimental bias results from previous studies, recommending the lottery equivalent probability approach (LEP). It involves asking questions seeking indifference in the decision maker's preferences between two sets of alternatives under uncertainty. For example, a lottery is presented where the decision maker can choose between a 50:50 chance for getting the worst value X^o or a particular value X_i, or a P_e chance for getting the best value X^* or $1-P_e$ chance for getting the worst value. A diagram often helps to visualize this problem.



The probability P_e is varied until the decision-maker is unable to choose between the two options. At this value, the utility for X_i can be determined easily by

$$U(X_i) = 2P_e$$

This directly follows from utility theory, which states that people make decisions in order to maximize their expected utility, or

[4] Ibid. (See chapter 4 for a discussion of vN-M single attribute utility functions.)

$$\max\{E[U(X)]_i\} = \max\left\{\left(\sum_j P(X_j)U(X_j)\right)_i\right\}$$

Once the single attribute utilities have been assessed, MAUA theory allows for an elegant and simple extension of the process to calculate the overall utility of multiple attributes and their utility functions.

There are two key assumptions for the use of MAUA.

1. *Preferential independence*
   That the preference of $(X_1^{'}, X_2^{'})\ \phi\ (X_1^{''}, X_2^{''})$ is independent of the level of X_3, X_4,…, X_n.
2. *Utility independence*
   That the "shape" of the utility function of a single attribute is the same, independent of the level of the other attributes. "Shape" means that the utility function has the same meaning up to a positive linear transformation, $U^{'}(X_i) = aU(X_i) \pm b$. This condition is more stringent than preferential independence. It allows us to decompose the multi-attribute problem into a function of single attribute utilities. (See derivation below for mathematical implications.)

If the above assumptions are satisfied, then the multiplicative utility function can be used to combine the single attribute utility functions into a combined function according to

$$KU(\underline{X}) + 1 = \prod_{i=1}^{n=6}\left[Kk_iU_i(X_i) + 1\right]$$

- K is the solution to $K + 1 = \prod_{i=1}^{n=6}\left[Kk_i + 1\right]$, and $-1<K<1$, $K\neq 0$. This variable is calculated in the *calculate_K* function.
- $U(\underline{X})$, $U(X_i)$ are the multi-attribute and single attribute utility functions, respectively.
- n is the number of attributes (in this case six).
- k_i is the multi-attribute scaling factor from the utility interview.

The scalar k_i is the multi-attribute utility value for that attribute, X_i, at its best value with all other attributes at their worst value. The relative values of these k_i give a good indication of the relative importance between the attributes—a kind of weighted ranking. The scalar K is a normalization constant that ensures the multi-attribute utility function has a zero to one scale. It can also be interpreted as a multi-dimensional extension of the substitute versus complement constant discussed above. The single attribute utility functions $U(X_i)$ are assessed in the interview.

If the assumptions are not satisfied by one or several attributes, the attributes can be redefined to satisfy the assumptions. (Many, if not most, attributes satisfy these conditions, so reformulation should not be too difficult.) Sometimes utility independence is not satisfied for several attributes. Several mathematical techniques exist to go around this problem. (For example, define aggregate variables made up of the dependent attributes. The aggregate variable is then independent. Nested multi-attribute utility functions can then be used in this case, with each function made up of only independent attributes.)

3.1.3 Derivation of multi-attribute utility function[5]

If attributes are mutually utility independent,

$$x = \{x_1, x_2, ..., x_n\}$$

$$U(x) = U(x_i) + c_i(x_i)U(\bar{x}_i) \qquad i = 1,2,...,n-1 \tag{1}$$

$\bar{x}_i$ is complement of x_i.

setting all $x_i = x_i^o$ except x_1 and x_j $j = 2,3,...,n-1$

$$U(x_1, x_j) = U(x_1) + c_1(x_1)U(x_j) = U(x_j) + c_j(x_j)U(x_1)$$

$$\frac{c_1(x_1) - 1}{U(x_1)} = \frac{c_j(x_j) - 1}{U(x_j)} \equiv K \qquad j = 2,3,...,n-1 \tag{2}$$

$$U(x_1), U(x_j) \neq 0$$

if $U(x_j) = 0$ → $U(x_1) = c_j(x_j)U(x_1)$ → $c_j(x_j) = 1$

from (2) above,

$$c_i(x_i) = KU(x_i) + 1 \quad \text{for all } i = 1,2,...,n-1 \tag{3}$$

Multiplying (1) out yields:

$$\begin{aligned} U(x) &= U(x_1) + c_1(x_1)U(x_2, x_3, ..., x_n) \\ &= U(x_1) + c_1(x_1)[U(x_2) + c_2(x_2)U(x_3, x_4, ..., x_n)] \\ &\quad \text{M} \\ &= U(x_1) + c_1(x_1)U(x_2) + c_1(x_1)c_2(x_2)U(x_3) \\ &\quad + \Lambda + c_1(x_1)\Lambda\, c_{n-1}(x_{n-1})U(x_n) \end{aligned} \tag{4}$$

Substituting (3) into (4)

$$\begin{aligned} U(x) = U(x_1) &+ [KU(x_1) + 1]U(x_2) \\ &+ [KU(x_1) + 1][KU(x_2) + 1]U(x_3) \\ &\quad \text{M} \\ &+ [KU(x_1) + 1]\Lambda\, [KU(x_{n-1}) + 1]U(x_n) \end{aligned} \tag{5a}$$

or

$$U(x) = U(x_1) + \sum_{j=2}^{n} \prod_{i=1}^{j-1} [KU(x_i) + 1]U(x_j) \tag{5b}$$

There are two special cases for equation (5b): where K=0, K≠0.

[5] Ralph L. Keeney and Howard Raiffa, Decisions with Multiple Objectives: Preferences and Value Tradeoffs, John Wiley & Sons, New York, NY (1976). (See pages 289-291.)

K=0:

$$U(x) = \sum_{i=1}^{n} U(x_i) \qquad (6a)$$

$K \neq 0$:

Multiply both sides of (5b) by K and add 1 to each.

$$KU(x) + 1 = \prod_{i=1}^{n} \left[KU(x_i) + 1 \right] \qquad (6b)$$

since $U(x_i)$ means $U(x_1^o, ..., x_{i-1}^o, x_i, x_{i+1}^o, ..., x_n^o)$, it can also be defined as

$$U(x_i) \equiv k_i U_i(x_i),$$

with k_i defined such that $U_i(x_i)$ ranges from 0 to 1. This function, $U_i(x_i)$, is the single attribute utility function.

Plugging this result into (6b) results in the multiplicative multi-attribute function used in B-TOS.

$$KU(x) + 1 = \prod_{i=1}^{n} \left[Kk_i U_i(x_i) + 1 \right] \qquad (7)$$

Since it was unlikely to be the case that the attributes did not have cross terms for utility, the utility team assumed that $K \neq 0$, and this equation is valid. Notice that it captures the tradeoffs between the attributes, unlike an additive utility function, such as (6a).

3.2 Process

This process aimed to design a space-based system using Multi-Attribute Utility Analysis (MAUA) to capture customer needs. Each architecture is measured by a set of attributes that are then mapped into a utility value. The architectures are then compared on the basis of utility for the customer and cost.

In general, the design of space systems starts with a point design that is usually provided by the customer. The MAUA process was used to evaluate many architectures. The attribute definitions are a mechanism for customer interaction and allow iteration of the definitions and expectations, and hopefully allow the designers to understand the underlying drivers of the customer's requirements. Once the design team has gained a deep understanding of the mission and the requirements on the performance of the system, the architectures are evaluated on the basis of their performance and cost. The choice of the architecture is therefore motivated by a real trade study over a large trade space.

This process has been chosen as a tool to decide the best architectures to perform the three customer defined missions (EDP, AOA and Turbulence missions). The objectives were to study the MAUA process and apply it for the first time to a space system design in order to choose the best family of architectures for a space-based ionospheric mapping system.

3.2.1 Comparison between the GINA process and Multi-Attribute Utility Analysis

3.2.1.1 GINA concept[6]

The A-TOS design project used the GINA process, developed by the Space Systems Laboratory, to make trade studies on possible architectures. The GINA method is based on information network optimization theory. The concept is to convert a space system into an information flow diagram in order to apply the optimization rules developed for information systems to space systems. This tool allows the design team to compare different architectures on the basis of performance and cost so as to be able to determine the best architecture(s).

The global process is the following:

- Define the mission objective by writing the mission statement
- Transform the system into an information network.
- Define the four Quality of Service metrics for the specific mission considered (signal isolation, information rate, information integrity, availability) so as to quantify how well the system satisfies the customer.
- Define the quantifiable performance parameters: performance, cost and adaptability.
- Define a design vector that groups all the parameters that have a significant impact on the performance or cost of the architecture. It represents the architecture tested.
- Develop a simulation code to calculate the details of the architecture necessary to evaluate the performance parameters and cost.
- Study the trades and define a few candidates for the optimum architecture.

3.2.1.2 GINA and MAUA

The methodology we followed is close to the GINA process since it aims at the same broad objective: evaluating architectures on the basis of a study over a huge trade space rather than around a point design.

MAUA offers more flexibility and can be more easily adapted to the specific mission studied. Instead of using the same performance parameters for all missions based on the information network theory, attributes that characterize what the customer wants are defined for the specific mission studied. Importantly, MAUA maps customer-perceived metrics (attributes) to the customer value space (utility). This allows for a better fit with the expectations of the customer. MAUA also offers a rigorous mathematical basis for complex tradeoffs. As in the GINA process, cost is kept as an independent variable and used after the trade space study to choose the best tradeoff between performance and cost.

MAUA has already been used in manufacturing materials selection and to help in airport design, but has not been applied to the design of complex space systems. The B-TOS project attempts to apply it to the design of a complex space system constellation.

[6] Shaw, Graeme B. The generalized information network analysis methodology for distributed satellite systems, MIT Thesis Aero 1999 Sc. D.

3.2.2 Detailed process

The first step consisted of defining the attributes. Attributes are the quantifiable parameters that characterize how well the architecture satisfies the customer needs (customer-perceived metrics). The attributes must be chosen carefully to accurately reflect the customer's wants for the system. Additionally, to truly characterize the system, the attributes should completely represent the system. (The attributes themselves are not unique, but instead should represent a non-overlapping subspace of characterization since they are the basis for making trades.) After defining the attributes, a utility questionnaire is developed. The questionnaire is then used in an interview with the customer to find the shape of his preferences. A follow-up validation interview corroborates the results and adds confidence. The multi-attribute utility function is derived from the interview results and represents the utility that the customer perceives from a given set of attribute values.

3.2.2.1 Preliminary definition of attributes

Early in the process, an initial list of possible attributes were defined for the specific mission we were studying. The following candidates for attributes were chosen:

- Mission completeness: to capture how well EDP measurements was performed.
- Spatial Resolution: to capture the importance of the distance between two consecutive measurements.
- Time Resolution: to capture the importance of the time delay between two consecutive measurements.
- Latency: to capture the effect of the time delay between the measurements to the user.
- Accuracy: to capture the impact of how precise is the measurements were; this was conceived as error bars on the EDP measurements.
- Instantaneous Global Coverage: to capture the issue of how much of the surface of the Earth was covered by the system.
- Lifecycle Cost: the issue was to capture the cost of the total mission from deployment to launch and operations over the 5 years of design lifetime.

These seven attributes were thought to capture the mission performance within our understanding of the mission at that point in the process.

3.2.2.2 Verification with the customer

The attributes have to be defined in collaboration with the customer and this is one of the crucial steps in the development of this method. Therefore, the preliminary definitions of the attributes were submitted to the customer to discuss any modifications. Most of the previously listed attributes were considered relevant and were kept in this first iteration.

3.2.2.3 Determination of the ranges

The customer was asked to provide a range for each attribute corresponding to the best case and the worst case. The best case is the best value for the attribute from which the user can benefit; a better level will not give him more value. The worst case corresponds to the attribute value for which any further decrease in performance will make the attribute useless. These ranges define the domain where the single attribute preferences are defined.

3.2.2.4 Iterative process to modify the attribute definition

The attributes have to describe customer needs accurately in order to meaningfully assist the trade study. Therefore, an iterative process is necessary to refine the list of attributes. This step has been a major issue in the B-TOS process.

First iteration:

Lifecycle cost was taken out of the attributes and kept as an independent variable that would drive the choice of the architecture at the end of the process. The first iteration was a discussion with the customer to come to an agreement on the definition of the attributes. The number of attributes drives the complexity and the length of the process and therefore, one goal was to minimize the number of attributes while still capturing all the important drivers for the customer. Mission completeness was suppressed because the instrument primarily drove how well the EDP mission was performed, which was not part of the trade.

Second iteration:

Our first understanding was that two missions were to be considered: EDP and Turbulence measurements. It appears that an additional mission was to be performed: Angle of Arrival measurements. The attributes were defined only for EDP measurements and so major modifications were required. The writing of the code had already been started and the aim was to minimize the modifications to the attributes. Only one attribute was modified: mission completeness. Mission completeness was reinstalled as a step function giving the number of missions performed. The customer gave us a ranking of the missions to help us define this function. EDP was to be performed, otherwise the mission was useless. The second most important mission was AOA, and last turbulence. So mission completeness was defined as: 0 for EDP, 1 for EDP/Turbulence, 2 for EDP/AOA and 3 for all three missions.

Third iteration:

Many issues emerged during the interview with the customer. Accuracy was left as EDP accuracy but it appeared to cause a problem. Accuracy was defined for EDP measurements but it became apparent that AOA accuracy was driving the accuracy of the whole system. EDP accuracy depends on the instrument, which is not traded, and on the error due to the fact that the satellite is still moving while taking measurements. The AOA mission requires a very accurate measurement on the order of 0.005 radians. This issue appeared during the interview. The first idea was to consider only the AOA accuracy since it was driving the system's accuracy but the AOA mission was not always performed. The second solution would have been to define a coupled single attribute preference curve but that was not possible because the two accuracies have very different scales. Finally it was decided that accuracy would have two different preference curves, one for EDP measurements and one for AOA measurements. If the AOA or turbulence missions were performed, AOA accuracy would apply, if only the EDP mission is performed, EDP accuracy would apply.

Moreover, the definition of the time resolution was refined. It was originally defined as the time interval between two consecutive measurements, however the customer had no real interest in this information. Instead, the customer wanted the time between two consecutive measurements at the same point in space. To capture this modification, the attribute was changed to Revisit Time. In essence, the design team was thinking in terms of a moving (satellite-centric) coordinate frame, while the customer was thinking in terms of a fixed (earth-centric) coordinate frame.

3.2.2.5 Development of the Matlab code

The Matlab code has as inputs the single attribute utility curves derived from the interviews and the corner point coefficients, k_i. The code is given a combination of values for the attributes and calculates the utility. The skeleton of the code was written before the interviews and the results of the interviews with the specific preferences of the customer were inputted as constants that modified the skeleton. Thus, the code is portable to utilize other customers' preferences.

3.2.2.6 Interview

The aim of the interview was to determine the preferences of the customer. Two different kinds of information are required to calculate the utility for every combination of values of the attributes:

- The single attribute preferences, which define the shape of the preference for each attribute within the worst/best range defined by the customer, independent of the other attributes. Below is an example of the single attribute preferences obtained from the interview. (Refer to Appendix B for the other attribute preference curves).

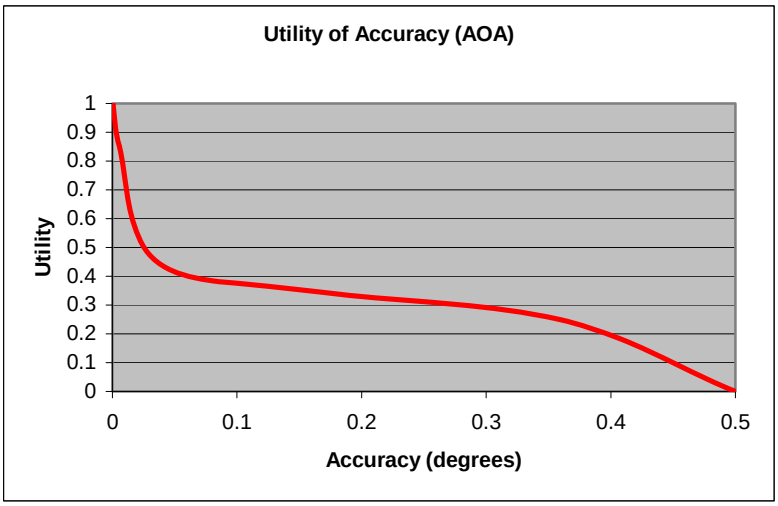


Figure 3-1 Single Attribute Preference Example

- The corner points, which allow a correlation between the single attributes and combinations of other attributes.

The probabilistic nature of the questions takes the issue of risk into account.

3.2.2.7 Validation Interview

The final step in the process was to check the consistency and the validity of the results of the first interview to ensure that the customer's preferences were captured. This was done during a second interview. In the B-TOS case, this interview was also used to check the assumptions of

the utility theory: preferential and utility independence. Assumption checking is usually done during the first interview, but time limitations pushed it to the second interview.

3.3 Initial Interview

The interview to ascertain the customer's utility took place on March 21, 2001. The aggregate customer, Dr. Bill Borer of the Air Force Research Laboratory (AFRL) at Hanscom Air Force Base, was present, in addition to Kevin Ray, also of AFRL. The entire utility team, consisting of Adam Ross, Carole Joppin, Sandra Kassin-Deardorff, and Michelle McVey, were also present. The presence of the entire utility team facilitated the decision process, as definitions and other questions could be changed or adapted by consensus following a brief discussion. Although the interview was expected to last two hours, it actually lasted approximately six hours.

The single attribute utility questions and questions to derive the corner points were prepared prior to the interview. These questions consisted of scenarios to descriptively explain possibilities in which different levels of a particular attribute might be obtained. The actual questions are attached in Appendix. Suggested attribute values between the best and worse cases (as defined by the customer) and suggested probabilities were included after the questions to fill in the blanks of the generic scenario. The suggested attribute values were those for which utility values would be measured. The suggested probabilities were ordered to facilitate bracketing in order to arrive at the indifference point. A worksheet followed each scenario and was used to record preferences at particular probabilities and the indifference point.

In addition to the questionnaire, an Excel worksheet was prepared for each attribute for real-time recording of the questionnaire responses. As the entries were made, the utility was plotted. This provided a redundant record as well as a means to signal the questioner when enough points had been collected on the curve. Each member of the utility team played a particular role during the interview. Adam asked the questions, Michelle recorded the results in the spreadsheet, and Sandra and Carole took the minutes and made observations.

The interview had a slow beginning, as each attribute definition had to be reviewed and the nature of the scenarios had to be explained. The probabilistic nature of the questions was unusual for Dr. Borer, so he developed his utility curve through discussions with Kevin Ray and Kevin translated by answering the lottery questions using his understanding of Bill's utility. Once this mechanism was adopted, the interview went smoothly. In addition, the interviewee was assured that there is no objectively "right" answer, as the utility must reflect their preferences.

We also asked the single attribute utilities and *k* values in a different order from that depicted in the interview in the Appendix. This was due to various miscommunications of attribute definitions or the learning curve associated with understanding the scenarios for some of the attributes. The order does not affect the results.

Significant changes or decisions made during the interview include the following:

1. The time resolution attribute was changed to revisit time.
   This was done to decouple the time attribute from the spatial resolution attribute. Dr. Borer had understood this to mean revisit time from the beginning and based his ranges on this assumption. Since the attributes must have a customer-perceived utility, we had to adapt the attribute to reflect the frame of reference of

the user. In this case, it was the frequency that a point in the ionosphere was measured and not a data set frequency.

2. Two accuracy attributes were adopted to capture the difference in both utility and type of accuracy required for the EDP and AOA missions.
   The accuracy requirements for the AOA mission were much more stringent than the EDP mission. In addition, the error bars as a percentage of the measurement used for EDP accuracy could not be used for AOA, as the origin of the angle was arbitrary. The EDP attribute utility would be used for those missions in which AOA was not conducted. For those missions that measured AOA, the AOA accuracy would apply. The questions were asked with AOA accuracy in mind. The EDP accuracy utility was scaled from AOA accuracy utility curve because they had the same shape.
3. The AOA accuracy range was 0.005 degrees (best) to 0.5 degrees (worst).
   This was later changed to 0.0005 degrees as the best case. The customer initially gave the ranges based on his assumptions of the technical limitations of the accuracy that could be achieved. He later found that the accuracy could be better. The utility curve was scaled using a linear transformation, which was valid because the customer was thinking in terms of best and worse cases possible, not specific numbers.

The attributes, their ranges and the *k* values are summarized in Table 3-1 below.

Table 3-1 Attribute Summary

Attribute	Definition	Best	Worst	k
Spatial Resolution	Area between which you can distinguish two data sets	1 deg X 1 deg	50 deg X 50 deg	0.15
Revisit Time	How often a data set is measured for a fixed point	5 minutes	720 minutes	0.35
Latency	Time for data to get to user	1 minute	120 minutes	0.40
AOA Accuracy	Error of angle of arrival measurement	0.0005 degrees	0.5 degrees	0.90
EDP Accuracy	Error of electron density profile measurement	100%	70%	0.15
Instantaneous Global Coverage	Percentage of globe over which measurements are taken in a time resolution period	100%	5%	0.05
Mission Completeness	Mission type conducted	EDP, AOA, and Turb	EDP only	0.95

3.4 Validation Interview

In order to establish preferential and utility independence, as well as validate the utility function derived from the original utility interview, a second interview was held on April 2, 2001. This interview was approximately 2.5 hours long. Attendees included Kevin Ray, Carole Joppin, Sandra Kassin-Deardorff, Michelle McVey, and Adam Ross. As Dr. Bill Borer was unable to attend, Kevin Ray acted as the aggregate customer. Although Dr. Borer is the actual aggregate customer, having Kevin Ray fulfill this role did not prove to be an issue because he had a clear idea of Dr. Borer's preferences.

Each of the utility team members was assigned a specific role during the interview. Adam conducted the interview, Sandra and Carole were assigned to take minutes and make observations, and Michelle recorded the answers. Although these were the assigned roles, many of the interview questions changed during the actual interview. This provided ample work for each of the utility team members, so the assigned roles do not properly reflect each of the member's roles during the interview. Although Adam still conducted the interview, the other three-team members spent most of their time either recording results or updating questions.

3.4.1 Utility Independence

The first set of questions, meant to establish utility independence, used a similar formatting as the original interview. Kevin Ray was asked to indicate his level of utility, using the lottery equivalent probability method, for a specific level of each individual attribute. Two sets of questions were asked using this format. One set was constructed with all of the other attributes at their best-case values and the other with the other attributes at their worst case values. Ideally, these two levels of utility should match, as the levels of the other attributes should not change the customer's utility for the attribute in question. The results are shown below.

Table 3-2 Utility Independence Results

	Initial Interview	Validation Interview	
Attribute	**Indifference Point**	**New***	**New$_*$**
Spatial Resolution	32.5%	32.5%	32.5%
Revisit Time	42.5%	37.5%	37.5%
Latency	37.5%	17.5%	22.5%
Accuracy (AOA)	42.5%	12.5%	12.5%
Accuracy (EDP)	42.5%	42.5%	42.5%
Inst. Global Coverage	48.0%	47.5%	42.5%
Mission Completeness	47.5%	48.0%	48.0%

New* = Indifference point for all other attributes at best performance values
New$_*$ = Indifference point for all other attributes at worst performance values

This table shows utility independence for all of the attributes. Each attribute had approximately the same level of utility associated with it regardless of the level of the other attributes.

The discrepancies lie in the information provided between the initial and validation interviews for the attributes AOA accuracy and latency. After reviewing the large discrepancy for AOA accuracy, it was decided the difference seen between the two interviews was probably due to the fact that a bracketing technique was used in the initial interview and was not used in the validation interview. In the initial interview, the bracketing process was started by comparing a mix of 0.16 or 0.5 degrees to 0.005 or 0.5 degrees. Kevin Ray indicated to the interviewer that he was not thinking about these numbers in absolute terms; he was thinking about them in terms of whether they were "good" or "bad." This is why it was important for the interview to utilize bracketing. By starting out with a relatively "bad" accuracy and increasing the accuracy in the next set of questions it is believed that Kevin Ray, the non-science customer, would be able to differentiate between the different levels of AOA accuracy. Thus, the interview can properly capture the relative "goodness" of the given accuracy. Although this process worked well in the initial interview, it was not used in the verification interview because of time constraints. Without this bracketing technique, it is believed that Kevin Ray saw the given accuracy values (0.01 or 0.5 vs. 0.005 or 0.5 deg) as "bad" and thus was willing to risk more to try to go for the better accuracy. Another issue with the bracketing vs. non-bracketing techniques is that the customer is much more likely to be concerned about being consistent with the bracketing case. Although Kevin Ray used the notes that he took from the initial interview to complete the validation interview, he would be less inclined to be consistent in the validation interview because he was only presented with one level of AOA accuracy instead of a series of accuracies. This error is due to the utility team's lack of interviewing experience and not the changing of customer preferences. It was also recognized that the customer was diligently trying to emphasize the importance of acquiring a high level of AOA accuracy.

The discrepancy in the preferences for latency between the initial and validation interviews is best attributed to human variability. Although the customer’s preferences may have remained constant between the interviews, his answers to the questions may change over time. Generally, the desire for self-consistency during the interview process actually helps the customer to solidify his preferences/beliefs. This is evident by looking at the other attributes, which remained relatively constant between the two interviews.

3.4.2 Preferential Independence

The second set of questions consisted of questions that asked for the customer’s preference between two combinations of two attributes, given that each of the other attribute levels remain constant. After asking a set of 12 questions of this format, the same questions were asked again (in random order) with the other attributes at a different level. (See Appendix for questions and results) These questions established preferential independence of all of the attributes.

3.4.3 Random Mixes

In addition to the utility and preferential independence questions, a set of questions were asked to determine the customer’s perceived utility for random mixes of varying levels of the attributes. These questions were done in a probability format similar to that used in the other parts of the interview. The primary difference was that the customer was asked to evaluate random mixes of the six attributes vs. the cases where all of the attributes are at their best and worst case values.

Table 3-3 Random Mix Results

Attribute Mix (spatial resolution, revisit time, latency, accuracy, instantaneous global coverage, mission completeness)	**Customer Estimated utility**	**Calculated Utility**
25x25, 5 min, 60 min, 80%, 45%, EDP	0.169384	0.64647
50x50, 2 hrs, 5 min, 90%, 30%, EDP	0.44463	0.75227
5x5, 30 min, 15 min, 0.005 deg, 55%, EDP/AOA/Turb	0.99999	0.99989
30x30, 4 hrs, 1hr, 0.25 deg, 30%, EDP/AOA	0.91469	0.95719
10x10, 6 hrs, 20 min, 75%, 95%, EDP	0.27525	0.58432
20x20, 40 min, 30 min, 0.5 deg, 60%, EDP/AOA/Turb	0.92931	0.98171

Table 3-3 shows the results of these questions. The random mix values do not correlate closely with the values calculated with the original multi-attribute utility function. These results most likely reflect the extreme difficulty, if not the impossibility, for a person to comprehend a 6-dimensional problem. The MAUA approach for capturing utility therefore plays a very useful role, allowing a person to look at a smaller dimension problem, which they can comprehend.

An important note is that when only the EDP mission was listed in the attribute mix, it was compared only to "best" and worst-case scenarios that only performed the EDP mission. This comparison was used because the customer values the AOA mission so highly that he would be willing to risk everything else for a small chance of getting that mission.

3.5 Lessons and Conclusions

3.5.1 Lessons learned about the process

- The number of attributes is an important factor in the process. The more attributes chosen, the longer the interviews and the harder for the customer to give valid answers while taking so many variables into account simultaneously. For the success of the process, the number of attributes has to be limited. Working with 6 attributes was already difficult and the interviews were long.
- The format of the questions in the interview is not straightforward and it may be difficult for the customer to capture the correlation between their needs and the risk percentages. The whole process is based on the determination of the preferences of the customer and it is crucial that the utility captured in the interviews reflect the customer's preferences.
- The interview to check the assumptions of the utility theory was carried out in a second interview. The questions could have been easily added to the first interview since they are of the exact same format. The customer is used to the questions and has his preferences clear in his mind during the first interview and it would be easier to properly check the independences.
- It seems difficult to check the validity of the utility by asking the preferences for a randomly chosen set of values for the attributes. The customer cannot clearly determine what the utility is for any set of 6 values.
- A major issue was the modifications of the attributes during the whole process and even during the interviews. The writing of the code had already begun while the attributes

were still changing. This was a major issue in the development of the code. It would have been helpful to complete the iterations of the attribute definitions before starting to write the code.

3.5.2 Refining the Process

The process was applied for the first time to the design of a complex space system. Here are some major concerns that came out of our study:

- For very complex missions, one of the major issues may be how to capture all of the trades in a small number of attributes.
- The customer is a central actor in the process and it may be difficult to implement this process with multiple customers that may have divergent opinions. Is it applicable to a mission (like an interplanetary mission) when the customer is not well defined?
- The format of the questions may also be an issue for ensuring that the design team captures the customer needs.

3.6 Conclusion

The application of MAUA to space system design appears promising. It provides a mathematical process to analyze a large trade space and can be adapted to the specific mission being designed. The major issue seems to be the number of attributes: complexity increases very rapidly with the number of variables. On the other hand it may be difficult to capture all of the trends of a complex system with a small number of attributes. Mathematical techniques exist within the theory to nest utility functions, which may result in the ability to partition interviews. More than six attributes can be measured in this way, while keeping the number of attributes in a given interview to a manageable level. Future work will be done in this area.

MAUA also proved very useful in deriving and analyzing the driving parameters for the architecture. (See Design Space chapter for more information.) All in all, MAUA proves to be a promising technique to synergistically combine with the GINA method.

4 Design Space

4.1 Overview

The purpose of this section is to document the rationale and decision making processes involved with the evolution of the design vector. This section will address what a design vector is, how it fits into the space system modeling element, how the sub-team and class developed it, what the final design vector variables are, and finally some lessons learned in the process.

4.2 Design Vector Development

The design vector is a critical element of the process, providing a means for considering a multitude of space system architectures. The design vector provides the fundamental (independent) variables that define the architecture trade space. In this class the design vector excludes model constants and focuses on those variables that have been identified to have significant impact on the specified mission design and hence play a role in determining the utility. It is important to note that since the variables are traded, rapid geometric growth of the trade space results, providing motivation to keep the list curtailed to only the key elements, while maintaining the ability to probe the utility of a wide variety of architectures.

The key steps in the design vector development process developed by the 16.89 class are:

1. Identify key system attributes from customer: The attributes provide the initial framework for synthesizing key variables and are discussed in Chapter 3.
2. Develop initial design variable list based on system attributes
    a. For a sub-team (3 members) to make use of available resources to create preliminary and/or modified lists
    b. Make use of QFD to pare down list and cross-check against attributes
    c. Discuss with full team and incorporate suggestions
    d. Iterate as necessary: a total of 9 iterations were performed
3. Provide final (or current best guess) design vector for model input

Step 1 is addressed in section 3.2 under the multi-attribute utility definition process. This section describes the processes associated with step 2 and explains the class results for step 3.

The class decided that in order to create an effective preliminary design vector definition process and successful iteration and updating, a specific design vector sub-team should be in charge of the process. The sub-team's responsibilities included:

1. Using all resources available, create an initial design vector based on the current understanding of the B-TOS missions and utility attributes.
2. Report these results to the full class and other system experts for iteration.
3. Iterate this process as necessary and maintain documentation of the entire process.

					RANGE										
					Units	%	%	deg., km	Hz	t	dB	%		$100M	
					ATTRIBUTES	Turbulence Mission Completeness	Global Survey Mission Completeness	Spatial Resolution	Time Resolution	Latency	Accuracy	Instantaneous Global Coverage	**TOTAL**	Lifecycle Cost	**TOTAL w/ COST**
						1	2	3	4	5	6	7		8	
	VARIABLES	Units	**CONSTRAINTS**	Weight											
1	Apogee Altitude	km	a > p		9	9	9	0	3	3	1	**34**	1	**35**
2	Perigee Altitude	km	a > p		9	9	9	0	3	3	1	**34**	1	**35**
3	Number of Planes	Integer			3	3	3	?	0	0	9	**18**	9	**27**
4	Swarm per Plane	Integer			3	3	3	?	0	0	9	**18**	9	**27**
5	Satellites per Swarm	Integer			3	3	9	1	0	0	1	**17**	9	**26**
6	Sub-Orbits per Swarm	Integer	concentric orbits									**0**		**0**
7	Size of Swarm	m			3	3	9	0	1	3	9	**28**	0	**28**
8	Sounding, [4]	Y/N			0	0	0	3	3	0	0	**6**	0	**6**
9	Number of Sounding Antennas	Integer	3 or 6		3	3	?	?	0	9	0	**15**	3	**18**
10	Short Range Communications, [4]	Y/N										**0**		**0**
11	Long Range Communications, [4]	Y/N			0	0	0	0	3	3	0	**6**	0	**6**
12	On-Board Processing, [2]	Y/N			0	0	0	0	3	3	0	**6**	0	**6**
13	Autonomy											**0**		**0**
	TOTAL				**33**	**33**	**42**	**4**	**16**	**24**	**30**		**32**	

Figure 4-1 QFD of Design Vector vs. Utility Attributes (iteration 2)

At the outset, one of the tools that were found to be effective in facilitating completion of these responsibilities was QFD. QFD, or Quality Function Deployment, was developed as a graphical technique to translate customer needs into parameters or attributes of the final product. Although QFD was developed for manufacturing and product design capabilities, the broad techniques and benefits of QFD were 'custom-fit' for the 16.89 systems development process. QFD provided the following benefits:

- Expedite correlation of variables with attributes
- Rank order most critical variables and influence on attributes
- Reduce variable list to minimize trade space dimensionality
- Minimize human biases
- Prioritize technical requirements
- Provide requirement and attribute trace ability and book keeping
- Provide a simple and easy to understand communication mechanism

The second iteration on the QFD matrix is shown in Figure 4-1 as an example. The vertical column contains the design vector test entries, which are the variables that are weighted against the attributes; the attributes are shown in the horizontal rows across the top of the matrix.

The QFD diagram in Figure 4-1 is in the developmental stage and is included so that the reader can gain a feel for the iteration process. It is interesting to compare the QFD iteration with the final design vector shown below. QFD provides a ready comparison of the test variables for the design vector by contrasting them against the list of attributes to determine relative weightings. Furthermore, a series of QFD spreadsheets such as this, in conjunction with the decision log forms, provides an excellent method of documentation and decision trace ability.

In order to iterate the QFD matrix, various resources were utilized by the design-vector sub-team and the full class. These include:

- A-TOS design code
- B-TOS interviews and attributes
- People: Bill Borer, MIT faculty and staff
- Tools: SMAD (general info, modeling equations, sample trades, etc.)
- Training: Prerequisite classes, undergrad info, etc.

The A-TOS design code served as a valuable starting point for evolving the design vector. The design variables from A-TOS were initially posed against the new attributes for the B-TOS projects and the most important variables were kept in the iteration process while those that were not important for this mission were eliminated. The resources above, particularity SMAD, the faculty and stuff, and personal experience of participants in the sub-team and class, were valuable in adding and testing new design variables. These were selected based on an understanding of the mission and the physics involved and then selecting which design variables appropriate to space system design should be included. Additionally, the customer specifically indicated interest in distributed functionality within a swarm, and thus indicated the need for design variables to capture this functionality trade. The sub-team iterated on the proposed design variables, often using QFD, to determine which variables would remain.

The sub-team and the entire class performed 9 iterations on the design vector to arrive at its final form. The process was repeated each time a new customer requirement, constraint, or change in the overall mission was introduced so that the team could ensure the proper design vector was maintained. Table 4-1 shows the final list of the design variables.

The results of the process used to create and maintain the design vector can be summarized as the following:

- The idea of breaking the design vector process down into a sub-team group was highly beneficial. This allowed a sub-set of the class to become familiar with the physics of the mission and the results of the attribute and utility interviews to the point where intelligent decisions about which variables to include and exclude could be made.
- The iteration process was critical with the full team and other experts. This allowed the sub-team to have confidence in the decisions that were made and to keep the class abreast as to why certain variables were included and others excluded. Although not often formally done, the full class modified the design vector list several times during large group design meetings. After each of these changes, the sub-group would update the new design vector.

- QFD served as a useful tool for deciding which variables were most important, as well as being a quick and easy way to document decision flow and design vector evolution.
- The process described in this section allows teams to create a design vector that is rigid enough to define unique architectures through model development, yet flexible enough to allow honing and modification with evolving attributes and mission requirements. This level of flexibility was shown to be of critical importance, because many changes were made throughout the process that influenced design variable choices.

4.3 Design Vector Variables

The resulting final design variables listed in Table 4-1 form the design vector. These variables form the basis of an "architecture" that is evaluated by the B-TOS code. This section defines each of the design variables and the rationale for inclusion in the design space.

Table 4-1 Final Design Variable List

	Variable	**Units**	**Rationale**
1	Apogee Altitude	Km	Specifies orbit/relationship to ionosphere
2	Perigee Altitude	Km	Specifies orbit/relationship to ionosphere
3	Number of Planes	INT	Key to meeting global coverage needs
4	Swarms per Plane	INT	Key to meeting global coverage needs
5	Satellites per Swarm	INT	Local coverage resolution
6	Size of Swarm	Km	Local coverage resolution
7	Number of Sounding Antennas	INT	Captures functionality trade
8	Sounding	0-3	Captures functionality trade
9	Short Range Communication	0-1	Captures functionality trade
10	Long Range Communication	0-1	Captures functionality trade
11	On-Board Processing	0-1	Captures functionality trade

Payload notation:
0: None
1: Send
2: Receive
3: Receive and Send

Other notation:
0: None
1: Yes (all)

INT: Integer value
km: kilometer

4.3.1 Apogee Altitude

Apogee altitude is measured in kilometers and is the maximum distance of a body in orbit from the center of the Earth. This variable was included because it specifies the orbit and its relationship to the ionosphere.

4.3.2 Perigee Altitude

Perigee altitude is measured in kilometers and is the minimum distance of a body in orbit from the center of the Earth. This variable was included because it specifies the orbit and relationship to the ionosphere.

In practice, both the apogee and perigee altitudes were set equal to one another, resulting in a circular orbit. Analysis of the mission resulted in no benefit to differing altitudes over the course of an orbit. A lower bound of 1100 kilometers was set by the customer to ensure the sounder is above the F2 peak of the ionosphere.

4.3.3 Number of Planes

The number of planes is an integer and specifies the number of unique orbital planes. This variable was included to drive the instantaneous global coverage and revisit time attributes.

4.3.4 Swarms per Plane

The number of swarms per plane is an integer and specifies the number of distinct swarms per orbital plane. A swarm is defined as a localized cluster of spacecraft operating in a synergistic fashion. A reference orbit defines the swarm orbit, with each spacecraft moving along perturbations of the reference orbit. In B-TOS each swarm had a center satellite moving in the reference orbit. This variable was included to drive the instantaneous global coverage and revisit time attributes.

4.3.5 Satellites per Swarm

The number of satellites per plane is an integer and defines the total number of satellites in a swarm. In B-TOS, each swarm was assumed to be identical. This variable was included to drive the spatial resolution and accuracy attributes.

4.3.6 Size of Swarm

The size of the swarm is measured in kilometers and specifies the radius of the Hill's ellipse for farthest satellite in the moving coordinate frame of the center satellite. The size specifies the structure of the swarm geometry, along with assumptions about configuration to perform the AOA mission. This variable was included to drive the accuracy attribute.

4.3.7 Number of Sounding Antennas

The number of sounding antennas could be 4 or 6 and is specified by the payload. Six antennas offer better data, but require more power. After continued discussion with the customer, B-TOS decided to fix the number at 6 for the sounders and 4 for the receivers. This variable was included to capture functionality trades.

4.3.8 Sounding

Sounding is a discrete variable, taking values of 0 to 3. Each number represents a discrete capability for the payload. 0:none, 1:send only, 2:receive only, 3:send and receive. None means no payload. Send only means only sounding. Receive only means only listening. Send and receive means sounding and listening. Value 1, send only, turned out to not make sense, so it was not used in the trades. Slight mass and power differences distinguished 2 from 3. The actual sounding design variable is a vector containing the sounding values for each of the satellites in the swarm. This allows for unique specification of each satellite. This variable was included to capture functionality trades.

4.3.9 Short Range Communication

Short-range communications is a discrete variable, taking a value of 0 or 1. 0: no capability, 1: send and receive. Originally this variable could take values of 0 to 3 like the sounding variable, but it was decided that only none or all capability made sense. Short-range communication is defined as intra-swarm, or within swarm, communication. Like the sounding variable, the short-range communication design variable is a vector containing the short-range communication values for each of the satellites in the swarm. This variable was included to capture functionality trades.

4.3.10 Long Range Communication

Long range communications is a discrete variable, taking a value of 0 or 1. 0: no capability, 1: send and receive. Originally this variable could take values of 0 to 3 like the sounding variable, but it was decided that only none or all capability made sense. Long range communication is defined as extra-swarm, or to TDRSS, communication. Like the sounding variable, the long-range communication design variable is a vector containing the long-range communication values for each of the satellites in the swarm. This variable was included to capture functionality trades.

4.3.11 On-board Processing

On-board processing is a discrete variable, taking a value of 0 or 1. 0: no processing, 1: “all” processing. At conception, this variable would have a range of discrete values representing varying levels of processing capability. For simplicity, the none or all split was used in B-TOS. No processing refers to no data processing capability. Necessary command processing capability is assumed on all spacecraft. “All” refers to processing capability necessary to reduce the uplink data rate by a factor of 3. Like the sounding variable, the on-board processing design variable is a vector containing the on-board processing values for each of the satellites in the swarm. This variable was included to capture functionality trades.

Several changes to the A-TOS design variables are readily apparent from the list of design variable choices. Foremost, the binary Mothership satellite choice has been eliminated, but the concept is maintained through appropriate selection of functionality. This means that it is possible achieve results that suggest for certain swarms no motherships may be required, whereas for other configurations 2 or 3 motherships might be most suitable. (Though, with distributed functionality, the definition of Mothership is blurry.) In addition, the swarm geometry is no longer a design parameter. It is assumed, so as to maximize the accuracy of the AOA mission. (Trades on drag have come to light that may require reinvestigation of these assumptions.) Variables 1-4 capture the large-scale constellation architecture trades, 5-6 capture the most important swarm-level architecture trades, and finally variables 7-11 capture the vehicle trades.

4.4 Conclusions

The purpose of this section was to document the rationale and decision making processes involved with the evolution of the design vector. This section addressed the definition and components of the design vector used in the generation of the B-TOS architectures. This chapter also included a brief overview of how the design vector fits into the space system modeling element, how the sub-team and class developed it, and the lessons learned in the process. Finally

the chapter concluded with a presentation of the final design vector and a description and rationale for each of the variables.

5 B-TOS Module Code Development

5.1 Overview

The critical factor in the B-TOS project was code development. Since the principal deliverable stated in our mission statement is the reusable code, capturing that objective was crucial. The B-TOS team had the advantage of being the second iteration of the TOS project; the team could inherit the work of the first iteration, A-TOS. Early in the process, the two members of the B-TOS team who were also members of the A-TOS team recognized the strong possibility of reusing the A-TOS code. In particular, since Adam was the integrator for A-TOS, he had a strong familiarity with the A-TOS code structure and understood the applicability to the B-TOS problem.

Once the B-TOS team had become familiar with the problem to be solved and modeled in B-TOS, Adam Ross held a seminar on the A-TOS code, covering execution, overall structure, and specific code details. The class also recognized the prospect for code reuse, though it did make sure to question each case of reuse to make sure unnecessary assumptions were not carried over from A-TOS. Thus, with two continuing members of A-TOS on the B-TOS team, knowledge and experience in the Matlab code writing and modeling process was readily continued.
The B-TOS project proceeded to build upon the foundation started in A-TOS and succeeded in expanding the functionality of the code and improving its theoretical underpinnings through the use of a utility function. Code-writing efforts were distributed to teams in order to divide the work and encourage parallel development. In this way, a "black-box" modular code design not only enabled the problem to be discretely manageable, but also resulted in a code that could be modularly upgraded. The "black-box" design kept the details of each module within each subgroup, with the integration team only concerned with the interfaces. The integration team created several tools that greatly streamlined the integration process, which was complicated by the nature of distributed module writing. (A-TOS did not have this problem since it was mainly written by three people who sat in the same room.)
In the end, the process worked well. In the face of changing customer requirements, the process held up well with minimum update efforts. The robustness of the code architecture allows for rapid adjustment of many of the design assumptions. It also allows significant flexibility for fidelity improvement. The overarching goal in the code development was to capture the basic functional relationships, while not precluding more detailed modeling to be installed at a later date. In this regard, the B-TOS code appears to be a success.

5.2 Development of Code Framework

In order to develop the architecture of the simulation code, the team took the following two steps. First, the team reviewed the A-TOS codes and learned its architecture. Based on that understanding, the team employed the Universal Modeling Language (UML) to develop the architecture for the B-TOS simulation model[7].

UML is a software development method for large software development efforts. It emphasizes understanding customer needs, requirement flow-down, decomposing the system to minimize

[7] Fowler, M. and Scott K., UML Distilled, second edition. Addison-Wesley, Boston, 2000.

integration problems, and visualizing the interactions among software modules. Three methods from UML were used in this project—the Use Case diagram, the Class Diagram, and the Sequence Diagram. They are shown in Figure 5-1,Figure 5-2, and Figure 5-3.

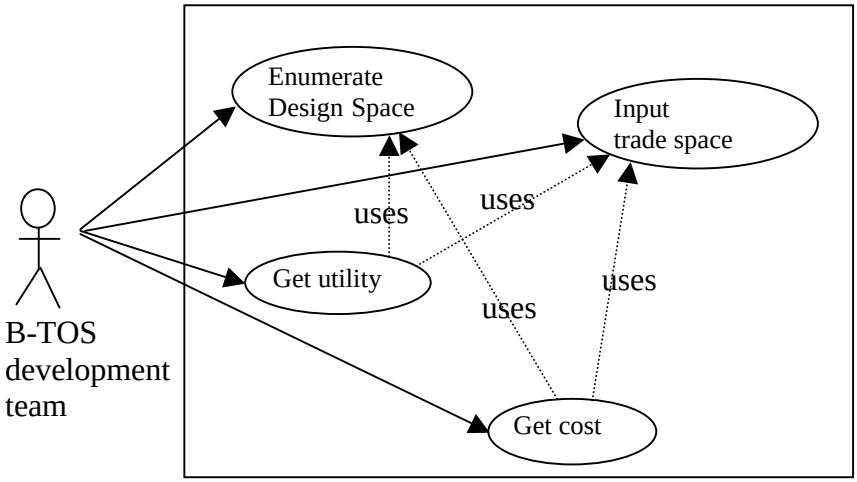


Figure 5-1 B-TOS Architecture Trade Software Use Case

In the Use Case diagram, the users of this software are the team itself. The purpose of the code was to develop a module in order to trade among different architecture choices based on their contribution to utility and cost.

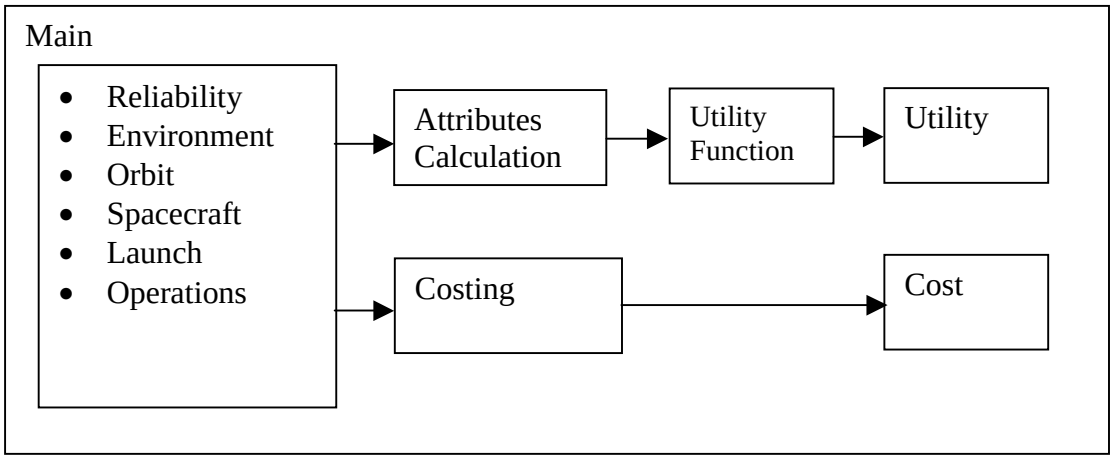


Figure 5-2 B-TOS Architecture Trade Software Class Diagram

The modules for B-TOS (Figure 5-2) were developed based on two principles. First, the team wanted to maximize the reuse of A-TOS code. Therefore, the structure of the A-TOS software

was assessed, and B-TOS software architecture was developed based on A-TOS code. Second, the software modules were design to be independent so that they can be easily integrated.

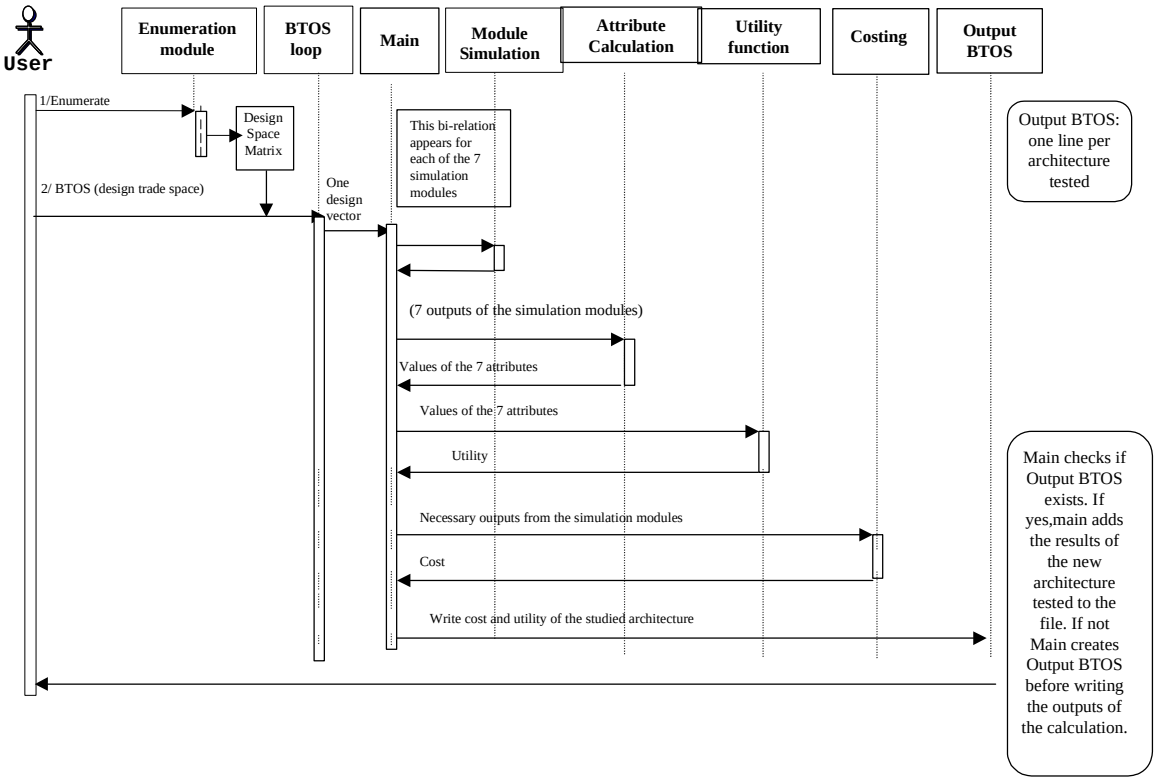


Figure 5-3 Sequence Diagram

After developing the modules, the sequence diagram (Figure 5-3) was constructed for the modules in order to depict the high level interactions among modules.

5.3 Organization Principle

After the architecture of the software was set, the class divided up into sub-teams to work on various modules in the software. The sub-teams were formed based on the software modules. Each module was assigned to at least two people in the class. One person was the primary representative of the module, with the other person as a backup. This setup was intended to avoid any single-point failures in the organization structure. Everyone in the class signed up for the modules in which he/she felt competent and was interested. In addition to the module teams, an integration team was also created to address the integration among the modules. The final organization structure was:

Table 5-1 Organization Structure for Code Development

Module	Primary Representative	Secondary Representative
Main	Adam Ross	Qi Dong
B-TOS	Adam Ross	Qi Dong
Orbit	Scott Kimbrel	Sandra Kassin-Deardorff
Environment	Sandra Kassin-Deardorff	Scott Kimbrel
Swarm	Nathan Diller	Brandon Wood
Spacecraft	Brian Peck	Nathan Diller
Launch	Dan Kirk	Brian Peck
Operations	Brandon Wood	Nathan Diller
Reliability	Dan Kirk	Michelle McVey
Costing	Michelle McVey	Dan Kirk
Attribute	Carole Joppin	Brandon Wood
Time	Carole Joppin	Nathan Diller
Utility	Adam Ross	Carole Joppin
Integration	Qi Dong Adam Ross	

5.4 Module Description Summary

There are nine major modules in the software. They are:

1. Swarm/Spacecraft Module
2. Reliability Module
3. Time Module
4. Orbit Module
5. Launch Module
6. Operations Module
7. Costing Module
8. Attributes Module
9. Utility Module

This section describes each module from the following six aspects:

- Introduction
- Required inputs
- Outputs descriptions
- Key assumptions
- Fidelity assessment
- Verification

5.4.1 Swarm/Spacecraft Module

5.4.1.1 Introduction

The swarm module populates the swarm by determining how many distinct spacecraft configurations are specified by the design vector, calling the spacecraft code once for each distinct configuration. The spacecraft code uses the functionality specification from the design vector to determine the mass, power and mean-time-to-failure for each spacecraft subsystem. This information is passed back to the swarm module which then creates several matrices (see output descriptions below) used by other modules (reliability, costing etc.). The swarm code is included in the *swarm.m* and *spacecraft.m* files.

5.4.1.2 Required Inputs

The swarm module takes inputs from the following modules:
DESIGN
CONSTANTS
The inputs are as follows:
DESIGN.swarm_matrix
DESIGN.long_range_comm.
DESIGN.short_range_comm.
DESIGN.sounding
CONSTANTS.lr_p_fail
CONSTANTS.lr_comm_mass
CONSTANTS.lr_comm_power
CONSTANTS.sr_p_fail
CONSTANTS.sr_comm_mass
CONSTANTS.sr_comm_power
CONSTANTS.payloadb_mass
CONSTANTS.payloadb_power
CONSTANTS.payload_b
CONSTANTS.sounding_p_fail
CONSTANTS.sounding_mass
CONSTANTS.sounding_power
CONSTANTS.receiving_p_fail
CONSTANTS.receiving_mass
CONSTANTS.receiving_power
CONSTANTS.payload_data_rate
CONSTANTS.cdh_with_processing_mass
CONSTANTS.cdh_with_processing_power
CONSTANTS.processing_p_fail
CONSTANTS.cdh_no_processing_mass
CONSTANTS.cdh_no_processing_power
CONSTANTS.no_processing_p_fail
CONSTANTS.number_of_gps_antennas
CONSTANTS.mass_per_gps_antenna
CONSTANTS.power_per_gps_antenna
CONSTANTS.number_of_magnetometers

CONSTANTS.mass_per_magnetometer
CONSTANTS.power_per_magnetometer
CONSTANTS.number_of_star_trackers
CONSTANTS.mass_per_star_tracker
CONSTANTS.power_per_star_tracker
CONSTANTS.number_of_sun_sensors
CONSTANTS.mass_per_sun_sensor
CONSTANTS.power_per_sun_sensor
CONSTANTS.number_of_torquers
CONSTANTS.mass_per_torquer
CONSTANTS.power_per_torquer
CONSTANTS.adacs_processor_mass
CONSTANTS.adacs_processor_power
CONSTANTS.number_of_engines
CONSTANTS.mass_per_engine
CONSTANTS.power_per_engine
CONSTANTS.isp_of_engine
CONSTANTS.number_of_thrusters
CONSTANTS.mass_per_thruster
CONSTANTS.power_per_thruster
CONSTANTS.eclipse_length
CONSTANTS.daylight_length
CONSTANTS.mission_life
CONSTANTS.max_solar_flux
CONSTANTS.max_sun_angle
CONSTANTS.solar_array_eff_direct
CONSTANTS.solar_array_eff_thru_batt
CONSTANTS.cell_specific_power
CONSTANTS.cell_degradation_per_year
CONSTANTS.cell_eff_range
CONSTANTS.inherent_degradation
CONSTANTS.battery_to_load_trans_eff
CONSTANTS.battery_DOD
CONSTANTS.battery_energy_density
CONSTANTS.subsat_density
CONSTANTS.bal_coef
ENVIRONMENT.Avgdelv

5.4.1.3 Output Descriptions

SWARM.distinct_sats:
Number of distinct satellites in the swarm.

SWARM.sc_matrix:
Matrix (dimension: distinct_sats by 6), where each column contains information about the following functions: sounding, processing, long-range communicating and short-range communicating. The last two rows of each column contain a unique identifier created for each distinct satellite and the number of satellites with that distinct functional configuration.

SWARM.sc_mass_matrix:
Matrix (dimension: distinct_sats by 3), where each column contains mass, power and number of satellites with that mass and power.

SWARM.sc_mttf_matrix:
Matrix (dimension: distinct_sats by 2), where each column contains mean time to failure and number of satellites with that mttf.

SWARM.sc_datarate_matrix:
Matrix (dimension: distinct_sats by 2), where each column contains data rate and number of satellites with that data rate.

SWARM.sc_subsystem_mass_matrix:
Matrix (dimension: distinct_sats by 10), where each column contains communications subsystem mass, payload mass, command and data handling subsystem mass, attitude determination and control subsystem mass, propulsion subsystem mass, power subsystem mass, thermal subsystem mass, structural mass, and propellant mass.
SWARM.tdrss_links:
Number of communications links between the swarm and the TDRSS communications satellites.
SWARM.software_cost:
Cost of the software needed by the swarm.

5.4.1.4 Key Assumptions

Fundamental equations
The equations in the spacecraft module may be found in the various chapters in SMAD dealing with subsystem design. Most are design "rules of thumb" or simple addition of specified constants, with the most notable exception being the calculations for the power subsystem. These are based upon the requirement that the batteries be able to provide peak power and that the solar arrays be able to provide average power for the duration of the mission. These equations account for degradation over the lifetime of the equipment.

Rationale for simplifications
The most glaring simplification in the spacecraft module is that the spacecraft is treated as a homogenous cylinder (mass evenly distributed throughout). This simplification was made to avoid having to fully design the spacecraft, since the architecture discrimination is much more important at this level. The rationale for this decision is that the cost model is only based upon mass and the volume is small enough that size should not be driving launch capabilities anyway. As such, it should have no impact on the architecture(s) chosen by the code.

Evolution of calculations
The calculations have remained essentially unchanged since first written. Much of the code remains unchanged even from the A-TOS spacecraft modules.

5.4.1.5 Fidelity Assessment

The swarm module populates the swarm with satellites and does so without making any assumptions about its layout. The spacecraft module is only as good as the relationships given in SMAD. As most of these relationships are approximations determined empirically from databases, they are inherently inexact. SMAD suggests margins of up to 20% when using these

relationships so early in the design process. As such, the fidelity of this code can be assumed to be no more than 80%.

5.4.1.6 Verification

The swarm and spacecraft modules were tested using a dummy design vector module and constants vector module. A wide range of functionality distributions were tested with emphasis on configurations that were likely to be a part of the final study.

5.4.2 Reliability Module

5.4.2.1 Introduction

This module uses a Markov Model to determine the probability of any of the B-TOS swarms being in any given state as a function of any time during the mission. For most cases considered, the code calculates for a single swarm, but the capability exists for considering multiple swarms. For the cases considered here, the module calculates reliability information at the being, middle and end of the mission life period. The module first considers all the satellite types and reads in those types along with the associated mean time to failure for each type. The code then reads in the number of each satellite type prior to the actual reliability calculation. To summarize, the input into the reliability calculation portion of the code is the number of satellites, the number of each type of satellite and the mean time to failure associated with each type. The code then uses the Markov Model (from A-TOS) to calculate the probability of each type of satellite being operational at any time during the mission. The module returns this set of probabilities as a matrix. The reliability code is contained within the *reliability.m, swarmrel.m*, and *MarkovModel.m* files.

5.4.2.2 Required Inputs

The reliability module takes inputs from the following modules:
DESIGN
CONSTANTS
SWARM
SWARMREL

The inputs are as follows:
DESIGN.swarms_per_plane
DESIGN.sats_per_swarm
DESIGN.number_of_planes
DESIGN.apogee_altitude
CONSTANTS.time_step
COSNTANTS.mission_life
SWARM.sc_mass_matrix
SWARMREL.mttf
SWARMREL.sats_per_tpe

The SWARM.sc_mass_matrix is a matrix of satellite masses and the number of satellites with that mass. This is used to calculate the mass-based spacecraft properties. The abbreviation mttf stands for mean time to failure and SWARMREL.mttf is a matrix where each mean time to

failure number is associated with an individual spacecraft type. This includes Mothership and Daughtership, as well as variations on the daughterships.

5.4.2.3 Output Descriptions

SWARMREL.steady_state_reliability:
This is a matrix that gives the steady state reliability numbers for all of the various satellite types in terms of a decimal percentage.
SWARMREL.working_sats:
This is a matrix that rounds the probability sums to give a final operational percentage of the number of satellites that are in operation at any given time during the mission.
RELIABILITY.P :
This is the probability matrix for each of the satellites

RELIABILITY.error:
This is an error flag that checks to ensure that the number of sub-satellites does not exceed the number calculated in the reliability module.

5.4.2.4 Key Assumptions

Fundamental equations
The Markov Model employed provides the reliability module with a continuous time state translation matrix for the model taking into account the number of satellites in the swarm. The Markov Model assumes that the swarms are always replenished to their full level when there are fewer than the full level of spacecraft remaining in the swarm. It also toggles between having and not having a mothership. This toggle can be done manually or can be turned off, with the mothership spacecraft parameters entering through the satellites per type matrix (current module version). If a mothership is present, or the mothership equivalent in the satellites per type matrix is present, failure of the mothership results in failure of the swarm. Currently, only up to 26 sub-satellites per swarm are supported, but this could be easily extended for future configuration studies. The code calculates the operational probability for each satellite type and takes a summation of these for each mission time.

Rationale for simplifications
Very few simplifications are made in this code, since the reliability module takes into account all the different satellite types and their corresponding mean times to failure. The simplifications that are made is that the model assumes that a swarm failure (loss of all sub-satellites or mothership) will be repaired through a re-launch.

Evolution of calculations
The most significant change in the reliability module was the ability to consider different types of satellites, with different numbers of each type in a swarm and, consequently, with different mean times to failure. This was accomplished by writing a new front-end modification to the code that would read in the satellite types, the number of each, and the associated mean time to failure. The code was also constructed so that these reliability calculations could be performed at any time during the mission. The beginning, middle and end of the mission were selected as the three representative times for analysis and comparison.

5.4.2.5 Fidelity Assessment

The fidelity of the reliability module suffered most from a lack of knowledge about the true mean time to failure of the various satellite types. Representative numbers were used for each type, but eventually these numbers will need to be improved based on the reliability of the mean time to failure of any critical components of the spacecraft. These numbers could then be easily inserted into the mean time to failure matrix for each spacecraft.

5.4.2.6 Verification

The reliability module was tested using various combinations of initial parameters, including varying the number of satellites (daughter and mother types), various numbers of each, a range of mean time to failures from 1 day to 10 times the mission length, various ranges on the mission life time, and studies to determine the minimum time step for the calculation. It was found that for most of the mean time to failures that were examined on the order of half to full mission length time, the degradation in the number of operational satellites was very small. However, significantly reduced mean times to failure did result in substantial loss of satellites and the need for replenishment. Plots of the number of operational satellites versus the mean time to failure for each type were generated so that when more accurate mean time to failure numbers are determined, a ballpark estimate of the system reliability could quickly be calculated. Finally, the variability to rounding up or down, when returning the final averaged probability for the system, and it was found that rounding up would give the more conservative probability value and hence was employed.

5.4.3 Time Module

5.4.3.1 Introduction

The time module was added to the code when reliability was implemented. It calculates the new mission performed by the system and different time delays for the calculation of latency for three moments during the mission. Those variables were initially calculated inside the swarm module, but since swarm and orbit were coupled, time was created to prevent a loop between orbit and swarm. The main steps of the program are detailed in the following paragraphs. The time code is contained within the *time.m* file.

5.4.3.2 Required Inputs

The time module takes inputs from the following modules:
DESIGN
CONSTANTS
SWARM
SWARMREL

The inputs are as follows:
CONSTANTS.proc_performance [in bits per second]:
Amount of data that can be processed per satellite with a processing capability
CONSTANTS.payload_data_rate [in bits per second]:
Data rate of the payload system that measures EDP, turbulence and angle of arrival
CONSTANTS.compression_ratio [number]:

Ratio of compression of the data characterizing the processing capability; it is defined as the ratio of the amount of data after processing over the amount of data before processing
CONSTANTS.telemetry_data_rate [in bits per second]:
Data rate for the telemetry subsystem (for the bus functions)
CONSTANTS.payload_b [in bits per second]:
Estimated data rate for the unknown payload
CONSTANTS.lrc_data_rate [in bits per second]:
Data rate for long-range communication
CONSTANTS.edp_time [in seconds]:
Total time to complete a sweep over all the frequencies for EDP measurements
CONSTANTS.phase_error_instrument [in degrees]:
Error in the determination of the phase due to the instrument error
CONSTANTS.gps_time_error [in seconds]:
Error in the determination of the time of arrival of a signal using the GPS system
CONSTANTS.c [in m/s]:
Speed of light
CONSTANTS.wavelength [in m]:
Wavelength chosen among the different wavelengths used for AOA measurements; used for the calculation of accuracy and ambiguity check
CONSTANTS.gps_pos_error [in m]:
Error in the position determination using the GPS system
CONSTANTS.minimum_suborbit_radius [in km]:
Lowest possible radius for a sub-orbit in a swarm
CONSTANTS.data_set_delay [in seconds]:
Delay between the end of a set of measurements and the next set
CONSTANTS.turb_time [in seconds]:
Time to complete a set of turbulence measurements
CONSTANTS.beacon_time [in seconds]:
Time to complete a set of angle of arrival measurements
CONSTANTS.earth_radius [in km]:
Earth radius
CONSTANTS.earth_mu [in km3/s2]:
Earth constant mu (=GM where G is the gravity constant and M the mass of the Earth)
CONSTANTS.no_tdrss_time [percentage, number between 0 and 1]:
Proportion of time on orbit when the swarm cannot see any TDRSS satellite
CONSTANTS.maintenance_time [percentage, number between 0 and 1]:
Proportion of time on orbit when the swarm cannot take any measurement because it is in maintenance mode
DESIGN.mission_to_task [number]:
Define the combination of missions (among EDP, Turbulence and AOA) that are realized by the swarm at the beginning of life
DESIGN.apogee_altitude [in km]:
Altitude of apogee of the swarm orbit
SWARM.sc_matrix [matrix, 5 rows, number of columns equal the number of different types of satellites in the swarm]:

This matrix summarizes the number of different types of satellites, giving the functionalities and the number of satellites within this category for each type (Row1: sounding capability, Row2: Processing capability, Row 3: Long-range communication capability, Row 4: Short-range communication capability, Row5: number of satellites in the swarm of this type)
SWARM.distinct_sats [number]:
Number of different types of satellites within the swarm
SWARMREL.working_sats [matrix, 3 rows, number of columns correspond to the number of different types of satellites in the swarm]:
This matrix gives the number of working satellites for each type of satellite at three different times during the mission (beginning of life, middle of the mission, and end of life)

5.4.3.3 Output Descriptions

TIME.working_sc [matrix, 7 rows, as many columns as there are different types of satellites in the swarm]:
This matrix gives the functionalities and the number of working satellites for each type of satellite in the swarm.
Row 1: Long-range communication capability
Row 2: Short-range communication capability
Row 3: Processing
Row 4: Sounding
Row 5: Number of satellites working at beginning of life
Row 6: Number of satellites working at middle of life
Row 7: Number of satellites working at end of life

TIME.receiving_working_sats [vector 3 components]:
This vector gives the total number of satellites in the swarm that can receive a signal at beginning of life, middle of life and end of life.

TIME.time_resolution_factor [number]:
Ratio of the data rate of the swarm and the maximum amount of data that can be compressed, taking into account the processing capability of the swarm. This is used in the calculation of time resolution.

TIME.no_edp_sats:
equals ERROR.no_edp_sats
Error flag needed by another module.

TIME.new_mission_to_task [vector, 3 components]:
This vector gives the new variable mission to task which represents the missions that the system is performing at the beginning of life, middle of life, and end of life

TIME.aoa_capability [vector, 3 components]:
This vector shows if the system is able to perform the AOA mission at the three instants (beginning, middle and end of life); (0: no AOA mission capability, 1: AOA mission performed)

TIME.time_resolution [s]:
This is the time resolution attribute that represents the time between the beginnings of two consecutive sets of measurements

TIME.period [s]:

Period of the swarm orbit

TIME.com_delay [s]:
Delay between two sets of measurements due to communication

TIME.proc_delay [s]:
Delay between two sets of measurements due to processing

TIME.sats_functions [matrix, 3 rows, 5 columns]:
Extracted from the variable called functionalities, which is not outputted. It gives the number of satellites in the swarm that are both receiving and sending (column1), are receiving (column 2) [all the satellites that can receive independently of sending capability], are processing (column3), have a long-range communication capability (column 4) and have short-range communication capability (column 5) for each of the three moments.

ERROR.time.no_edp_sats [binary]:
Check if EDP is measured. The error flag is set at 1 if no EDP measurement is done.

ERROR.time.no_lrc [binary]:
Check if there is at least one satellite with a long-range communication capability in the swarm.

5.4.3.4 Key Assumptions

Data Flow

Processing is assumed to be only a compression of the payload A data. The code assumes a maximum amount of data that can be compressed depending on the processing capability of the swarm: the processing system has a constant performance. The telemetry data rate is set as a constant, independent of the number of working satellites in the swarm. Different constants are used, such as the compression performance, compression ratio, telemetry data rate and payload B data rate.

New Mission to Task

The code assumes that:

- EDP mission is feasible if at least one satellite can send and receive in the swarm.
- AOA mission is feasible if at least one satellite at least can receive in the swarm.
- Turbulence mission is feasible if one satellite can send and one satellite can receive in the swarm. Those two functionalities can be done by the same or different satellites.

In addition to these constraints, there must be at least one satellite with long-range communication capability in the swarm.

Ambiguity Check

The ambiguity check has major assumptions that are explained in the next paragraph. The code uses a configuration of the satellites on the sub orbits of the swarm in order to calculate the minimum number of satellites necessary to fill the swarm. In this configuration, there is a satellite in the center of the swarm. A wavelength and a constant for the instrument phase error were chosen for the calculation. The minimum radius for the sub orbits of the swarm was set as a constant value defined in the constant vector.

Time Resolution

Processing capability has been accounted for in time resolution by multiplying time resolution by a factor called the time resolution factor. This factor captures the added delay if the swarm data rate exceeds the long-range communication capacity.

Time Delays for Latency

Processing delay is set as a constant equal to 0. The percentage of the orbit dedicated to maintenance and the percentage of orbit when TDRSS is not in view are set as constants.

Algorithms

Functionality

The program first summarizes what the functionalities of each type of spacecraft are and the number of satellites of each type. The TIME.working_sats is an extension of the SWARM.sc_matrix incorporating reliability and degradation of the system over time. The 4 first rows of SWARM.sc_matrix provide the different functionalities of each type of spacecraft and the last three rows of the matrix outputted by the reliability module provide the number of working satellites for each type of spacecraft.

Another matrix is built to evaluate some capabilities at swarm level. Functionality summarizes the total number of satellites within the swarm that are sending, receiving, sending and receiving, have a long range communication subsystem, have a short range communication subsystem, have a processing capability. This will be used throughout the program to evaluate the performance of the swarm and the capability of the system to perform the different missions.

Data Flow

The evaluation of the data flow is used to determine the delay due to communication and, therefore, the trade-off on processing capability. This is accomplished in the calculation of the time resolution via a time resolution factor (see time resolution paragraph). Using the functionalities of the swarm, the total payload data rate and the compression capacity of the swarm are computed; some payload A data rate and compression performance are assumed and then multiplied, respectively, by the number of payloads and processors in the swarm. The amount of data that cannot be compressed is just the difference between the total amount of payload data and the compression performance of the swarm, or the maximum amount of data that can be processed in the swarm. The data rate after compression is then computed as the compressed data divided by a compression ratio that has been estimated. From these, the swarm total data rate is the sum of the telemetry data rate and the payload B data rate (non-payload A data rates are, therefore, not compressed), the data rate after processing for the data compressed, and the extra data from payload A that could not be processed.

The parameter that is used in time resolution is the time resolution factor. It aims to consider that the more processors that are present in the swarm, the better the compression and, therefore, the shorter the delay before a new set of measurements can begin. The factor is defined as 1 if the swarm total data rate is greater than the long-range communication capability, and as the ratio of the swarm data rate over the long-range communication capacity otherwise. This captures the additional communication delay present if the swam cannot process all of the data because the processing capability is too low.

New mission to task

The new mission to task matrix is initialized as the variable mission_to_task from the design vector, which is the missions performed at the beginning of life. To compute what the missions will be with a degraded system, what the system can do at each of the three snapshot moments is first calculated. A matrix called mission_possible (matrix with 3 rows and 3 columns]) summarizes whether the system can perform EDP, AOA and turbulence missions at each of the three representative moments of the mission. Each row represents a moment in the mission; the columns represent each mission. This is determined by testing the different requirements for each mission. 0 means that the mission is not feasible; 1 that the mission is feasible. The new mission to task is then determined by what the system can do at the time considered and what the system was doing at the previous time.

Ambiguity check

The determination of the angle of arrival is influenced by the accuracy of the measure but also by the degree of ambiguity. One of the constraints of the system is to have no ambiguity. The distributed space system is used as an interferometer for the AOA mission and ambiguity is linked to how well the swarm is filled. The main notion involved is the notion of baselines or distances between pairs of satellites. Ambiguity on measurements from satellites on an outer ring is eliminated by the satellites in the consecutive inner ring if the number of satellites in the inner ring is sufficient and if they are at a certain distance from the satellites in the outer ring. The detailed calculation of the ambiguity constraint is developed below.

The code checks if the system matches the constraint of zero ambiguity, in other words, if there are enough satellites to fill the swarm. The idea is to calculate the minimum number of satellites required to fill a swarm with the radius defined in the design vector. If the number of working satellites is larger than the minimum required number of satellites the swarm is filled and the architecture is declared valid. Otherwise, the architecture cannot complete AOA measurements and the new mission to task is updated.

Time resolution

Time resolution depends on the missions performed. It is defined as the sum of the time required to perform each of the missions to be performed (set by new mission to task) and a constant delay, where the total sum is multiplied by the time resolution factor that accounts for the delay due to communication if the swarm data rate exceeds the long range communication capacity.

The time to perform turbulence and AOA missions are set as constants. The time to complete EDP measurements depends on the number of satellites able to perform EDP measurement, since the frequencies are split over the different satellites. The time to perform EDP measurements is the total time to complete a sweep over all the frequencies divided by the number of satellites that can do EDP measurements.

Time resolution is computed at each of the three snapshot moments during the mission and, therefore, accounts for the degradation of the system.

Time Delays for Latency

Two time delays are computed for the latency calculation: communication delay and processing delay.

Two phenomena are assumed to contribute to communication delay: the time when the system is not in view of TDRSS and the maintenance time. The total delay over one orbit corresponds to the period of the swarm on its orbit multiplied by the percentage of the orbit spent in

maintenance or out of view of TDRSS. This delay is then transformed into the delay per set of measurements by dividing the total delay by the number of sets of measurements performed during one orbit. Processing delay is currently a constant set at 0.

Fundamental equations

Ambiguity check

The configuration used for the ambiguity calculation is a triangle with three satellites per ring in a swarm.

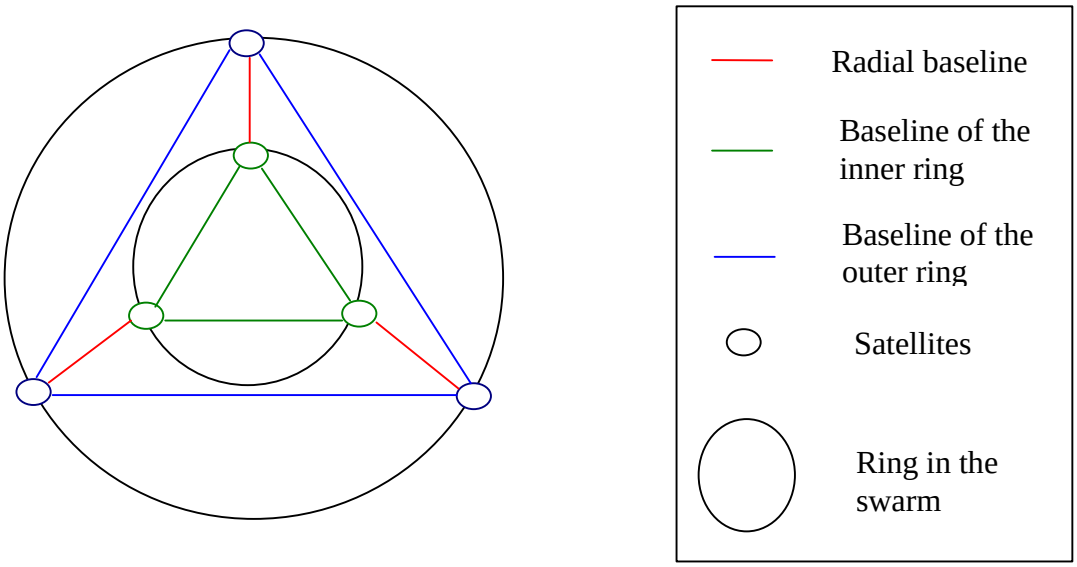


Figure 5-4 Swarm configuration for ambiguity criteria

Figure 5-4 illustrates the configuration chosen to calculate the ambiguity criteria. The two reasons for this choice of configuration are the following:

- There are three satellites per ring because that is the minimum number of satellites needed to have a three-direction determination of the angle of arrival.
- The satellites are placed at the vertices of an equilateral triangle. This configuration may not seem optimal at first because one of the aims to resolve ambiguity is to have more different baselines. However, in order to have one baseline resolve the ambiguity on the measurements performed by the satellites forming the previous baseline, the baselines have to be parallel. So the blue and green baselines have to be parallel pairs.

The radial baselines represented in red in the figure are not used in the calculation.

The criterion to resolve ambiguity is expressed as a constraint on consecutive baselines. If the different baselines are sorted from the smallest one to the biggest one, the criterion to have the inner ring resolve the ambiguity of the outer ring is the following:

$$\frac{D_{bigger}}{D_{smaller}} \leq \frac{1}{N}$$

where:
D_{bigger} is the biggest baseline among the two baselines compared, and $D_{smaller}$ is the smallest baseline among the two baselines compared.

N characterizes the maximum ratio between two consecutive rings in the swarm necessary to resolve ambiguity. This number is derived from interferometry relations:

$$N = \frac{\Phi}{2\pi}$$

where Φ is the total phase error; the sum of the phase errors due to an error in position determination, an error in time determination, and an error due to the instrument.

$$\Phi = \Phi_{position} + \Phi_{time} + \Phi_{instrument}$$

The minimum number of satellites is then computed:

- Starting with the outer ring, which has as its radius the radius of the swarm defined in the design vector, the smallest radius possible for the next inner ring is calculated. The radius matching exactly the criterion defined previously.

$$D_{smaller} = D_{bigger} * N$$

- The radius obtained for the inner ring is compared to the minimum ring radius. This minimum has been arbitrarily set so that satellites will not collide. If the radius is too large, then the process is iterated and a new ring is added inside the swarm. If the radius is too small, the process stops. The number of rings is the minimum number of rings necessary to fill the swarm, which means having zero ambiguity in the measurement. The last ring that falls below the limit is suppressed. The satellite that is in the middle of the swarm resolves the ambiguity on the last inner ring. To have a sufficient baseline in the center, booms may be added to the center satellite.
- The minimum number of satellites to fill the swarm is therefore:

$$n_{\min} = n_{rings} * 3 + 1$$

where:
$n_{\min}$ is the minimum number of satellites to fill the swarm
n_{rings} is the number of rings determined by the iteration

This number of satellites is then compared with the number of receiving satellites, which are the satellites useful for AOA measurement. If the number of receiving satellites is large enough, the system can perform the AOA mission, otherwise new mission to task is modified. This calculation is done at each of the three times of the mission to account for the degradation of the system.

Rationale for simplifications
Ambiguity Check
A simple configuration was used to calculate if the ambiguity was resolved or not. The rationale for the choice of the configuration is explained above. The ambiguity issue has been simplified

by assuming that the ambiguity of the last ring could be resolved by a single spacecraft in the center of the swarm, with a boom if necessary.

Evolution of calculations

DATA FLOW

The data flow calculation was added later on in the code to add fidelity to the time resolution calculation.

New mission to task
This has been implemented with reliability. The first version did not take into account the ambiguity issue. After the ambiguity check, a second part was added to adapt new mission to task if the AOA mission is not feasible because of ambiguity.

Ambiguity check
The ambiguity check has been a much-iterated issue. There are two main versions:

- The first idea was to use the configuration of the swarm determined by the ORBIT algorithm. From this assumed partition of the satellites, ORBIT computed the coordinates of the different baselines and sorted them from the smallest to the biggest. TIME was then testing if the number criterion was verified for each of the consecutive baselines. The tests of the code showed that this criterion was a significant driver and that the concept appeared to be wrong. One contradiction was that for a given radius of the swarm, if two configurations were considered, one with 3 sub-orbits matching the number criterion and the second with one more sub-orbit in addition to those same three sub-orbits, the second one did not match the number criterion. The model was indicating that the second architecture could not resolve ambiguity, while the first one could. This has no physical explanation since the second one had at least the same capability as the first one, since it has the same sub orbits and the same swarm radius.
- The ambiguity issue was reworked to derive the new model explained previously.

5.4.3.5 Fidelity Assessment

Data Flow
The use of the time resolution factor is a rough generalization of how to take processing the data into account. A better and more precise model of processing and specifications of processors could improve fidelity.

Ambiguity Check
The ambiguity has been a very iterative process. Fidelity has been increased. It may be interesting to iterate the ambiguity calculation again, and in particular to change the process. Instead of assuming a configuration of the satellites on the rings and calculating the minimum number of satellites in the swarm necessary to resolve ambiguity, an alternative would be to optimize the configuration of satellites for resolving ambiguity.

The calculation is done in the module for a frequency, and for the baseline defined by the swarm radius. Fidelity could be added by computing the baselines from STK. It was not done because computation time was excessive.

Time Resolution

Fidelity can be improved by more accurately modeling processing capability. Also, autonomy has not been considered in the model because no quantitative algorithm has been found with which to implement it into the simulation code.

Time Delays for Latency
Fidelity can be added by modeling processing delay.

5.4.3.6 Verification

Time Test Module
A test module simulating the time module inputs was used to separately test the syntax of the time module before integrating it to the B-TOS module.
Case Study for New Mission to Task
A special study has been performed in order to be sure that the new mission to task vectors were correctly computed. Each time variable was separately tested with different combinations of mission to task and reliability numbers.

Ambiguity Check

The ambiguity check was also studied separately to determine what the enumeration of the trade space would be. The ambiguity resolution drove the choice of the portion of the trade space to be tested, since AOA was one of the most important criteria for the customer. For the run, the swarm radii were chosen so that they would cover the whole range of the accuracy attribute. The attribute depends on the total phase error and the swarm radius (because the accuracy is determined by the maximum baseline which is calculated from the swarm radius). In an Excel spreadsheet, the number of sub-orbits, accuracy, and the radii of each sub-orbit were derived from the swarm radius and the assumed instrument phase error, in order to select the appropriate swarm radii to include in the tradespace.

5.4.4 Orbit Module

5.4.4.1 Introduction

In this module, two-body propagation is used over one day and orbit maintenance is assumed. J2 and J4 perturbations are not used. The module propagates a Walker constellation of swarms. There is one sub-plane per swarm and logarithmic spacing is used between the sub-orbits (or "rings") of the swarm. The swarms are configured to project a circle in the horizontal plane.

The swarm configuration consists of one center-satellite and three sub-satellites per sub-orbit. This configuration was used to create the baselines necessary to measure beacon angle of arrival data and to create the fill factor to eliminate ambiguity. The orbit code is contained within the *orbitprop.m* and *swarmorbits.m* files.

5.4.4.2 Required Inputs

The orbit module takes inputs from the following modules:
DESIGN
CONSTANTS
TIME

The inputs are as follows:

DESIGN.perigee_altitude
DSEIGN.apogee_altitude
DESIGN.number_of_planes
DESIGN.sats_per_swarm
DESIGN.radius_of_swarm
DESIGN.swarms_per_plane
CONSTANTS.subplanes_per_swarm (=1 for B-TOS)
CONSTANTS.arg_perigee (=0)
CONSTANTS.inclination
CONSTANTS.earth_radius
CONSTANTS.earth_mu
CONSTANTS.propagation_time_secs
CONSTANTS.propagation_steptime_secs
CONSTANTS.walker_interplane_spacing
CONSTANTS.walker_raan_spread
CONSTANTS.propagate_only_centersat (0 or 1)
TIME.time_resolution
TIME.no_edp_sats

5.4.4.3 Output Descriptions

The outputs of swarmorbits.m are as follows:

SWARMORBITS.swarmsat:
A matrix of the orbital elements for each satellite, including apogee and perigee altitudes, inclination, argument of perigee, RAAN, and true anomaly.

ERROR.swarmorbits.anyerror:
Equals 1 if there are any errors in swarmorbits.m, otherwise zero.

ERROR.swarmorbits.satsperswarm_is_1:
Equals 1 if true, otherwise zero.

ERROR.swarmorbits.subplanes_lessthan_satsperswarm:
Equals 1 if true, otherwise zero.

The outputs of orbitprop.m are as follows:

ORBIT.average_revisit_time:
Average revisit time for a grid of points; the grid is based upon the spatial resolution.

ORBIT.spatial_resolution:
The nadir angle swept out by the center satellite during t = time resolution.

ORBIT.instant_percent_global_cov:
Percentage of grid covered in t = time resolution; does not include polar regions north and south of latitude 65 degrees because grid currently does not extend to those regions.

ERROR.orbitprop.error_from_swarmorbits:
Equals 1 if an error is output from swarmorbits.m, otherwise zero.

ERROR.orbitprop.satsperswarm_morethan_26:
Equals 1 if true, otherwise zero.

ERROR.orbitprop.no_edp_sats:
Equals 1 if there are no working EDP satellites, otherwise zero.

5.4.4.4 Key Assumptions

Fundamental equations
The orbital parameters for each of the satellites in the swarm must be determined in order to provide the proper inputs to STK. The original swarm projects a vertical 2:1 ellipse along the global orbit. The ionospheric mapping mission requires distinct measurements distributed horizontally over a segment of the ionosphere. We decided to project a circle in the horizontal plane with a radius equal to the semi-major axis of the ellipse. The individual satellites must be given cross-track elements relative to the reference orbit at the center of the swarm. These incremental differences in orbital parameters are derived from the geometry of the swarm and uniquely describe the orbit for each satellite. These parameters include the following:

- Delta RAAN (Right Ascension of the Ascending Node)
- Delta Inclination
- Delta perigee
- Delta apogee
- Delta argument of perigee
- Delta true anomaly

The spatial resolution is defined as a conical angle originating at the center of the Earth and is determined by the time resolution (time between data sets) and the orbital velocity. The spatial resolution projects a circle on the surface of the Earth. The effective field of view (FOV) is a conical angle that originates at the center of the swarm and projects the same size circle on the Earth's surface. The FOV is used in STK to calculate revisit time and global coverage statistics.

Rationale for any simplifications
The average delta V's for station-keeping due to atmospheric drag were found to be small at the altitudes considered, so a constant was used in the spacecraft module. It was later determined that for large swarm radii, the delta V requirements for formation-keeping in the outer sub-orbits can be large due to J2 effects. This could be alleviated by not projecting a horizontal circle, at least for the outer sub-orbits. This sensitivity analysis has been done for some frontier architectures, but not for the entire tradespace.

The effective field of view was utilized to emulate an optical system so that the coverage and revisit statistics could be calculated by STK.

Evolution of calculations
The module was developed using the A-TOS code as a baseline. Since the number of sub-orbits per swarm was not a design variable in B-TOS, the logarithmic spacing calculation was not used in the same manner. The number of satellites per swarm constrained the number of sub-orbits by placing one satellite at the center and three in each succeeding sub-orbit. Discrete sets of satellite numbers were then considered.

5.4.4.5 Fidelity Assessment

The module used STK to ensure high fidelity orbit trajectories. This required a Matlab-STK interface.

5.4.4.6 Verification

Extreme cases were tested in order to test the assumptions. Visual inspections of the swarm geometry in three-dimensional STK animations were also used to verify the configuration.

5.4.5 Launch Module

5.4.5.1 Introduction

This module selects the lowest cost launch vehicle that can deploy all of the satellites in a single swarm using the appropriate launch vehicles as a function of the number of satellites per swarm, the mass per satellite, the stowed dimensions of a satellite, the orbital altitude, the launch vehicle mass capacity, and the launch vehicle payload fairing dimensions. Once a launch vehicle has been selected, the total cost for initial deployment is computed. The launch code is contained within the *launch.m* file.

5.4.5.2 Required Inputs

The launch module takes inputs from the following modules:
CONSTANTS
DESIGN
SWARM

The inputs are as follows:
DESIGN.swarms_per_plane
DESIGN.sats_per_swarm
DESIGN.number_of_planes
DESIGN.apogee_altitude
CONSTANTS.stowed_height
CONSTANTS.side_length
CONSTANTS.LV_name
CONSTANTS.LV_Cost_Dim_Matrix
CONSTANTS.LV_Perfromance_Matrix
SWARM.sc_mass_matrix

The variable sc_mass_matrix is a matrix of satellite masses and the number of satellites with that mass. This is used to calculate the mass-based spacecraft costs. The LV notion stands for Launch Vehicle, and the Cost Matrix contains the following information:

LV_Cost_Dim_Matrix
Fairing Dimensions Matrix
Rows Launch Vehicle Types
Column 1 Cost
Column 2 Fairing Diameter
Column 3 Fairing Height

A complete description of the launch vehicles, including dimensions (fairing diameter and fairing height), as well as cost, can be found in the constants module. The options considered were: Pegasus XL, Taurus, Athena 2, Athena 3, and Delta II launch vehicles. The Launch

Vehicle Performance Matrix contains a series of altitude that the mass of the payload is to be launched to with a range of 200-1500 km.

5.4.5.3 Output Descriptions

The outputs from the launch module are final code outputs, and thus are not inputs into any other modules. The outputs are as follows:

LAUNCH.LV_Capacity_Matrix:
Maximum number of spacecraft each launch vehicle can deploy in a single launch

LAUNCH.One_Plane:
This describes the launch vehicle suite for a single plane.

LAUNCH.LV_Selection_Initial Deployment:
This variable describes the suite of launch vehicles that is required for deployment of the initial constellation.

LAUNCH.Launch_Cost_Initial_Deployment:
This variable stores the initial launch cost for a given system.

ERROR.Launch_No_LV_Suitable:
Flag checks to ensure that the spacecraft fit into the available payload fairing.

5.4.5.4 Key Assumptions

Fundamental equations
This model makes use of the average satellite mass to calculate the launch vehicle selection criteria. This is a good approximation for launch vehicle selection, sizing, and cost considerations. The satellite density and volume are calculated using a typical density given in SMAD, used for estimating volume. The stowed height is calculated using a cylindrical shape profile. The code then calculates, using the total mass and volume, whether the series of spacecraft are within specifications to be launched to the selected altitude. The deployment cradle increases the launch mass by 25%.

Rationale for simplifications
This model makes use of the average satellite mass to calculate the launch vehicle selection criteria. This is a good approximation for launch vehicle selection, sizing and cost considerations. The module currently makes the assumption that all of the launches are completely successful, in that there is no failure rate or risk modeling done. However, this could easily be appended by adding a probability of failure or reliability model. This would be handled by either assuming a constant rate of failure (easiest method to employ) or by utilizing a Markov Model similar to the model used in the Reliability Module. If the constant failure rate is modeled it would be included by increasing the launch cost by that fraction.

Evolution of calculations
The launch module has remained quite similar to the A-TOS code in its logical progression. The changes have included the ability to incorporate an average spacecraft mass for the preliminary calculations, and to perform launch calculations for any given type of spacecraft in the swarm. The team decided to use the average spacecraft mass version (average spacecraft mass was weighted by the number of motherships and daughterships) of the module for the preliminary

runs, since the variability was not that large and this would allow for the most expedient way to arrive at useful results and trends. This set of calculations was then checked by an Excel spreadsheet for the frontier architectures.

5.4.5.5 Fidelity Assessment

The costing module is only as accurate as the launch vehicle data that could be found, as well as some rounding error associated with the actual altitude at which the spacecraft arrives. Launch site is not taken into account, and as was stated above, the failure rate of the launch vehicles was also not considered. The fidelity of the model is also somewhat compromised for expediency by using an average spacecraft mass to perform the costing and launch vehicle selection calculations. However, a version of this module does exist which allows the user to perform a launch and costing analysis for any of the individual spacecraft types. Its output should be used as a tool for comparing the relative sets of launch vehicles as well as their costs, rather than as an absolute set of launch conditions or cost number. The error bars on the spacecraft mass range from 5-20% depending on the difference between the average spacecraft mass and the maximum deviance of a single spacecraft mass. This error would be largest if there are a large number of daughterships and one wants to consider the launch parameters and costing for a mothership.

5.4.5.6 Verification

The launch module was tested under numerous average spacecraft masses to ensure that a suitable launch vehicle could be selected over a range of possible spacecraft masses. To test this, the maximum spacecraft mass, as well as the minimum spacecraft mass, were tested to ensure that the launch vehicles could launch these two representative masses. Furthermore, the average spacecraft mass was calculated as a weighted average, and this number was checked in the code for each configuration. The code was tested to ensure that both cylindrical and cubic satellite configurations could be placed in the launch vehicle. Launch vehicles, configurations and costs were calculated for various final orbital altitudes. A spreadsheet was set up to ensure that the results of the module were reasonable for all the frontier architectures. This spreadsheet checked the launch cost for the maximum and minimum mass spacecraft and then ensured that the calculated cost is indeed the mass weighted average.

5.4.6 Operations Module

5.4.6.1 Introduction

This module calculates the cost of operations by using spacecraft quantity and reliability data to size the required workforce. Learning curves are used on each of the seven different types of personnel to account for increasing personnel capability as the operations team gains experience throughout the mission lifetime. The cost of the required facilities is calculated, while segregating the startup and recurring expenses. The output variables are sums of different components of these cost structures. The operations code is contained within the *operations.m* file.

5.4.6.2 Required Inputs

The operations module takes inputs from the following modules:
DESIGN
CONSTANTS
SWARM
SWARMREL

The inputs are as follows:
DESIGN.swarms_per_plane
DESIGN.number_of_planes
SWARM.tdrss_links
SWARMREL.steady_state_reliability
CONSTANTS.checkout_ratio
CONSTANTS.staffed_shifts
CONSTANTS.satellites_controller
CONSTANTS.pay_rates
CONSTANTS.turnover_rate
CONSTANTS.train_hours_skill
CONSTANTS.ojt_ratio
CONSTANTS.group_train_scale
CONSTANTS.engineer_learning_curve
CONSTANTS.minimum_engineering
CONSTANTS.maximum_engineering
CONSTANTS.orbitanalyst_learning_curve
CONSTANTS.tasks_plan
CONSTANTS.plans_satellite_day
CONSTANTS.time_task
CONSTANTS.unconflicted_tdrss_access
CONSTANTS.planner_learning_curve
CONSTANTS.manager_ratio
CONSTANTS.hardware_maint
CONSTANTS.software_maint_ratio
CONSTANTS.overhead_ratio
CONSTANTS.computer_cost
CONSTANTS.cubicle_cost
CONSTANTS.connectivity_cost
CONSTANTS.floorspace_person
CONSTANTS.construction_cost
CONSTANTS.leasing_cost
CONSTANTS.facility_maintenance_cost
CONSTANTS.additional_nonrecurring_cost
CONSTANTS.additional_recurring_cost
CONSTANTS.ops_scale_factor
CONSTANTS.ops_plot_flag
CONSTANTS.ops_output_flag
CONSTANTS.mission_life
CONSTANTS.tdrss_link_cost

CONSTANTS.no_tdrss_time
CONSTANTS.shift_duration
CONSTANTS.mission_type
CONSTANTS.connectivity_annual_cost

5.4.6.3 Output Descriptions

The outputs from the operations module are a series of cost structures that integrate into the costing module. In addition, the operations module produces a matrix of labor statistics useful for quantifying the size and ability of the operations workforce. The following table lists the components of this matrix.

Row (labor type)	Column (labor data)
Controllers	Pay Rate ($/hr)
Engineers	Turnover Rate (fte/yr)
Support	Training Time (hrs)
Orbit Analysts	Post-launch Checkout Daily Work (hrs/day)
Mission Planners	Normal Operations Daily Work (hrs/day)
Trainers	Annualized Cost ($/yr)
Managers	Total Labor Cost ($)
Overhead	

The output variables are as follows:
OPERATIONS.total_mission_ops_cost
OPERATIONS.annual_ops_cost
OPERATIONS.nonrecurring_costs
OPERATIONS.recurring_costs
OPERATIONS.labor

5.4.6.4 Key Assumptions

Rationale for simplifications
The costing module is based upon the small spacecraft cost estimating relationship.
The fundamental premise for the simplifications in this module is that labor costs account for the majority of operations costs for a space system. Facility and computer costs are included but the modeling accuracy emphasis remains on the labor calculations. In addition, the operations center cost model assumes an entirely new center must be constructed with a devoted operations staff. In reality, operations facilities would probably be acquired from previous space missions, and operations personnel might migrate between multiple space missions. Since this dynamic would be challenging to model accurately, and since the results would be very specific to the organization that actually operated the space mission, it was not incorporated into the B-TOS model.

Modern operations center design focuses heavily on reducing space mission costs through increased use of autonomous control in both the space and ground segments. The effects of satellite autonomy are modeled by reducing the number of spacecraft the operations center is responsible for observing and controlling. The number of spacecraft is dependent on the number of TDRSS links required to operate the space segment. This, in turn, relates to the number of swarm motherships, since each mothership has the space-to-ground TDRSS communication package on board.

Evolution of calculations
The operations module has a highly modified evolution chain that begins with the TechSat21 code developed in MIT's Space Systems Laboratory. In the fall of 1999, another class used the TechSat21 operations module code as a baseline for its operations module in a similar space systems design process. David Ferris, a graduate student in that class was responsible for this major revision to the operations module. He later updated the code for A-TOS, the first design iteration of this space mission, in the winter of 2000-2001. This A-TOS code was slightly modified to account for different reliability and spacecraft inputs for B-TOS.

5.4.6.5 Fidelity Assessment

Adequate modeling of the impact of space segment and especially ground segment autonomy are the most significant calculations absent from this module. In addition, a number of the constants used to calculate costs were unavailable or questionable. Most notably, these included the cost of continuous access to TDRSS and the cost of ground software development and maintenance. The model does, however, account for labor training, turnover, and varying workloads as the mission progresses through its operational life. The numbers used for these calculations were derived from direct operational experience in U.S. Air Force space operations facilities.

5.4.6.6 Verification

The operations module output was verified by comparing test cases against first hand operational experience. This served to verify the learning curve assumptions and labor data. The facility construction values for the different test cases also matched anticipated results.

5.4.7 Costing Module

5.4.7.1 Introduction

This module uses a loop to calculate the spacecraft costs; integration, assembly, and test costs; ground support equipment costs; and program level costs, including learning curve effects. It adds these costs to the costs of launch, operations, and software to come up with a total lifecycle cost. The code also calculates the errors associated with the spacecraft costs; integration, assembly, and test costs; ground support equipment costs; and program level costs. The costing code is contained within the *costing.m* file.

5.4.7.2 Required Inputs

The costing module takes inputs from the following modules:
DESIGN
CONSTANTS
SWARM
LAUNCH
OPERATIONS

The inputs are as follows:
DESIGN.swarms_per_plane
DESIGN.sats_per_swarm
DESIGN.number_of_planes

DESIGN.apogee_altitude
CONSTANTS.learning_curve_slope
CONSTANTS.Lifecycle_Cost_Plot_Flag
CONSTANTS.Recurring_Non_Recurring_Costs_Plot_Flag
CONSTANTS.van_allen_alt
CONSTANTS.rad_hard_scale_factor
SWARM.sc_mass_matrix
SWARM.software_cost
LAUNCH.Launch_Cost_Initial_Deployment
OPERATIONS.total_mission_ops_cost
OPERATIONS.annual_ops_cost
OPERATIONS.Nonrecurring_Costs
OPERATIONS.Recurring_Costs

All of the costs listed above are self-explanatory except the sc_mass_matrix. It is a matrix of satellite masses and the number of satellites with that mass. This is used to calculate the mass-based spacecraft costs.

5.4.7.3 Output Descriptions

The outputs from the costing module are final code outputs, and thus are not inputs into any other modules. The outputs are as follows:

COSTING.Non_Recurring_Lifecycle_Cost:
This includes spacecraft, launch and non-recurring operations costs.

COSTING.Recurring_Lifecycle_Cost:
This includes recurring operations costs and replenishment costs. (See simplifications section below)

COSTING.Total_Lifecycle_Cost:
This includes all spacecraft, operations, and launch costs.

COSTING.TFU_Spacecraft_Cost:
This is the theoretical first unit spacecraft cost.

COSTING.Spacecraft_Cost:
This is the total cost of all spacecraft hardware.

COSTING.Operations_Cost_Lifecycle:
This is the total lifetime operations cost.

COSTING.Spacecraft_Cost_Lifecycle:
This is the total cost of spacecraft hardware, ground support equipment, program level costs, and integration, assembly, and test.

COSTING.Launch_Cost_Lifecycle:
This is the total cost of all launches.

COSTING.total_cost_error:
This includes error on spacecraft, ground support equipment, program level costs, and integration, assembly, and test.

ERROR.costing.sat_mass_out_of_range:
Equals one if out of range, zero otherwise.
ERROR.costing.input_of_0_for_num_sats:
Equals one if out of range, zero otherwise.

5.4.7.4 Key Assumptions

Fundamental equations
The costing module is based upon the small spacecraft cost estimating relationship (CER) in Space Mission Analysis and Design, 3rd ed. (p. 797-799, 809), which is solely based on mass and a learning curve factor. It is valid for spacecraft between 20-400 kg. All of the spacecraft that we considered were well within this range.

Rationale for simplifications
The final output for the costing module neglects replenishment costs. This assumption was made in order to facilitate the use of previously developed launch and reliability modules. Although the B-TOS iteration of the code does not consider these costs, the costing module does have the functionality to do so. If the launch and reliability modules were updated to calculate the launch costs associated with replenishing satellites, it would simply require removing the comments on a few lines in the costing module to incorporate these costs.

Evolution of calculations
The function has basically remained the same since first written. The most significant change is the addition of costs for radiation hardening.

5.4.7.5 Fidelity Assessment

The costing module is only as accurate as the CER that was used. Its output should be used as a tool for comparing the relative costs of different architectures, rather than as an absolute cost number. The error bars on the spacecraft costs range from approximately 20-40% of the overall spacecraft costs. This error increases with decreasing satellite mass and increased number of satellites.

5.4.7.6 Verification

The costing module was tested under both extreme and normal conditions to verify its output. It was tested with and without learning curve savings (i.e. with only 1 satellite of each type or multiple satellites of each type), and it was run with a wide range of spacecraft masses.

5.4.8 Attributes Module

5.4.8.1 Introduction

The *calculate_attributes.m* module calculates the value of the six attributes for the specific architecture tested and accounts for reliability and the degradation of the system by calculating those attributes at three different times during the mission: beginning of life, middle of life, and end of life. All the attributes are vectors with three components, one for each of the three instants in the mission at which the architecture is evaluated.

- **Spatial Resolution**

The spatial resolution is an output of the orbit module and no calculation is required in this module. In order to calculate EDP accuracy, we need the distance on the surface of the Earth covered by the center of the swarm between the beginnings of two consecutive sets of measurements. This is computed from the spatial resolution. Since the spatial resolution is the angle measured from the center of the Earth between these two data sets, the spatial resolution as a distance on the surface of the Earth is obtained by converting the angle to radians and then multiplying by the radius of the Earth.

- **Accuracy**

  The algorithm to calculate the accuracy strongly depends on the type of mission: EDP and AOA accuracies were so distinct that we had to use two different algorithms. Therefore, accuracy is composed of two variables: EDP accuracy and AOA accuracy.

- **EDP Accuracy**

  The EDP accuracy is calculated from payload data given by Bill Borer. Those data provide tables of EDP accuracy as a function of the spatial resolution as a distance on the surface of the Earth. EDP accuracy is given by the equation that would fit best those data.

- **Beacon Accuracy**

  The accuracy for the angle of arrival mission has been more difficult to develop. It is based on interferometry considerations between the satellites of the swarm. The detailed equations are developed below. AOA accuracy depends on an error in the determination of the phase of the signal. This error has three different origins: the error in the position determination by GPS, the error in the time determination with GPS, and finally the phase error due to the measurement instrument.

- **Latency**

  Latency is the sum of all the time delays between the measurements to the delivery to the user. It includes the time resolution (time for measurement and time to process the data before taking new measurements), communication delay, processing delay and ground operations delay. All the time delays added are either constants or calculated in the time module. All the time delays are defined in seconds, so the sum has to be translated into minutes so as to be consistent with the utility function.

- **Revisit Time**

  The Orbit Module calculates revisit time in seconds from STK. Revisit time has to be converted from seconds to minutes to be consistent with the utility function.

- **Global Coverage**

  The Orbit Module calculates global coverage with STK. Again, coverage has to be transformed from a percentage between 0 and 100 into a number between 0 and 1 to be consistent with the utility units.

- **Mission Completeness**

  Mission completeness is based on the variable new_mission_to_task calculated by the time module.

5.4.8.2 Required Inputs

The attributes module takes inputs from the following modules:
CONSTANTS
DESIGN
SWARM
SWARMREL

The inputs are as follows:
CONSTANTS.earth_radius [in km]:
Earth radius

CONSTANTS.utility.spatial_res [matrix, two columns, 6 rows]:
This matrix gives the value of the attribute tested during the interview with the customer in the first column and the corresponding single attribute utility in the second. It is used to test if the calculated attribute for the specific architecture we are computing is within the range defined by the customer.

CONSTANTS.utility.accuracy_EDP [matrix, two columns, 6 rows]:
Same form as previous constant.

CONSTANTS.c [in m/s]:
Speed of light

CONSTANTS.bearing [in radians]:
This is the angle between the line normal to the plane of the swarm and the beacon. As the satellite moves this will be continuously changing, but for the purpose of weighing separate swarms this is set as a constant in the module.

CONSTANTS.gps_time_error [in seconds]:
Error in the determination of the time of arrival of a signal using the GPS system

CONSTANTS.gps_pos_error [in meters]:
Error in the position determination using the GPS system

CONSTANTS.wavelength [in m]:
Wavelength chosen among the different wavelengths used for AOA measurements and used for the calculation of accuracy and ambiguity check

CONSTANTS.phase_error_instrument [in degrees]:
Error in the determination of the phase due to the instrument error

CONSTANTS.ground_delay [in seconds]:
Delay in the delivery of the data to the user due to ground operations

CONSTANTS.utility.accuracy_AOA [matrix, two columns, 6 rows]:
This matrix gives the values of the attribute tested during the interview with the customer in the first column and the corresponding single attribute utility in the second. It is used to test if the calculated attribute for the specific architecture we are computing is within the range defined by the customer.

CONSTANTS.utility.latency [matrix, two columns, 6 rows]:
Same form as previous constant.

CONSTANTS.utility.revisit_time [matrix, two columns, 6 rows]:

Same form as previous constant.

CONSTANTS.utility.global_coverage [matrix, two columns, 6 rows]:
Same form as previous constant.

CONSTANTS.utility.mission_complete [matrix, two columns, 6 rows]:
Same form as previous constant.
DESIGN.radius_of_swarm [in km]:
Radius of the outer sub-orbit of the swarm

ORBIT.spatial_resolution [in seconds]:
Distance between two sets of measurements; distance traveled by the center of the swarm during the time resolution
ORBIT.average_revisit_time [in seconds]:
Time between two measurements of the same point in the ionosphere

ORBIT.instant_percent_global_cov [in percentage, number between 0 and 100]:
Percentage of the surface of the Earth covered during a time resolution period
TIME.time_resolution [vector of three components, in seconds]:
Time between two sets of measurements

TIME.com_delay [in seconds]:
Delay in the delivery of the data due to communication through TDRSS

TIME.proc_delay [in seconds]:
Delay in the delivery of the data due to on-board processing

TIME.new_mission_to_task [vector of three components]:
From the initial mission to task defined in the design vector, this vector gives the missions performed by the swarm at three different periods (beginning of life, middle of life, and end of life), accounting for the degradation of the system.

5.4.8.3 Output Descriptions

The *calculate_attributes.m* module outputs two structures: ATTRIBUTES and ERROR. The ATTRIBUTES structure gathers the values of the different attributes at the three different times during the mission, while ERROR collects the error flags used to trace attributes that would fall outside the range defined by the customer. The following section presents each output with a brief description.

ATTRIBUTES.spatial_resolution [in degrees]:
(same as ORBIT.spatial_resolution) Angle measured from the center of the Earth between the positions of the center of the swarm at the beginning of two consecutive sets of measurements.

ATTRIBUTES.edp_accuracy [in percentage, number between 0 and 1]:
Value of the accuracy of EDP measurement (see the calculation below)

ATTRIBUTES.accuracy [in percentage between 0 and 1 or in radians]:
It is equal to beacon accuracy if AOA mission is performed. Otherwise, it is equal to the EDP accuracy.

ATTRIBUTES.beacon_accuracy [in radians]:
Accuracy of the measurement of the angle of arrival of the beacon signal

ATTRIBUTES.latency [in seconds]:
Time delay between the measurement of the data and the delivery to the end user

ATTRIBUTES.revisit_time [in seconds]:
Time between two consecutive measurements of the same point in the ionosphere

ATTRIBUTES.global_coverage [percentage, number between 0 and 1]:
(transposed ORBIT.global_coverage in the right range) Percentage of the surface of the Earth covered during a time resolution period (meaning between the beginning of a set of measurements and the beginning of the next one)

ATTRIBUTES.mission_completeness [number between 0 and 4]:
Missions completed by the system

ATTRIBUTES.attribute_values [matrix]:
This matrix gives the value of the attributes at three different periods in the mission.

ERROR.attribute.spatial_resolution_out_range
ERROR.attribute.accuracy_out_range
ERROR.attribute.latency_out_range
ERROR.attribute.revisit_time_out_range
ERROR.attribute.global_coverage_out_range
ERROR.attribute.mission_completeness_out_range

5.4.8.4 Key Assumptions

Fundamental equations
The physics behind the calculation of the AOA accuracy is one of the most important criteria for the customer and was one of the main points of iteration in the development of the code. The AOA accuracy was one of the main issues in B-TOS.

AOA accuracy is calculated from interferometry theory. The accuracy is calculated from the phase error:

$$d\theta = \frac{\lambda}{2\pi D} d\varphi_{total}$$

where $d\theta$ represents the accuracy, λ the wavelength of the signal emitted by the beacon on Earth (the beacon realizes a sweep over various frequencies, but for the comparison between different architectures we chose one frequency and therefore, one wavelength), D is the maximum baseline (distance between two satellites in the swarm) and $d\varphi_{total}$ is the total phase error.

The phase error has three components:

- Due to error in position determination (dD) (related to GPS system error) φ_{pos}

$$\varphi_{pos} = \frac{2\pi D}{\lambda \cos(\theta)} dD$$

- Due to error in time determination (dT) (related to GPS system error): φ_{time}

$$\varphi_{time} = \frac{2\pi c}{\lambda \cos(\theta)} dT$$

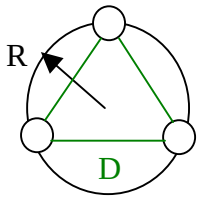
- Instrument phase error: φ_{inst} (set as a constant depending on the performance of the instrument). Therefore:

$$d\varphi_{total} = \varphi_{pos} + \varphi_{time} + \varphi_{inst}$$

$$d\theta = \frac{c}{D\cos(\theta)} dT + \frac{1}{\cos(\theta)} dD + \frac{\lambda}{2\pi D} d\varphi_{inst}$$

Rationale for simplifications
Spatial_resolution (See Orbit Module)

Accuracy
- We did not consider any turbulence accuracy because the two primary missions that were driving customer preferences were EDP and AOA missions.
- For dT and dD, we took the usual values for a GPS system since we assumed that every satellite had a GPS system.
- The calculation was done for D equal to the maximum baseline since the maximum baseline is responsible for accuracy while the smaller baselines contribute to reducing and eliminating ambiguity in the signal. The ambiguity issue was addressed in the B-TOS code in the time module.
- We assumed that the maximum baseline was constant. To determine it we assumed a configuration where three satellites are on the outer sub-orbits in the swarm:



R: outer radius
D: maximum baseline

So we took: $D = R\sqrt{3}$

The rationale for such a configuration is addressed in the description of the Time Module in the explanation of the ambiguity issue.

Revisit_time (See Orbit Module)

Global_coverage (See Orbit Module)

Evolution of calculations
Accuracy
The accuracy calculation has been a much-iterated process.
- In the first iteration of the code, accuracy was exclusively EDP accuracy until we realized during the utility interview that the customer valued the EDP and AOA missions equally.

The scales of the two accuracies were completely different and could not be wrapped together in a single accuracy variable that would have been a weighted sum of EDP and AOA accuracies; this would not have been valid for the utility function theory. It appeared that when the AOA mission was performed, the AOA accuracy was driving the total accuracy of the system, since AOA accuracy was far much sensitive to the system than EDP accuracy (which is determined mainly by the instrument). Therefore we decided to calculate both EDP and AOA accuracies. When AOA was performed, accuracy would be the AOA accuracy; otherwise accuracy would be the EDP accuracy.

- In the first calculation of AOA accuracy, we assumed that the instrument phase error would be negligible, which gave us the previous equation for accuracy, but only with the two terms linked to GPS (position and time). The calculation gave us such good accuracies, that they were out of the range defined by the customer.
- In the last iteration, we considered the three terms and realized that we could not neglect the instrument phase error.
- Minor modifications were made to the formula; a modification in the position phase error (to account for the error in position in the right direction) and an absolute value to all the error terms.

5.4.8.5 Fidelity Assessment

Spatial resolution (See Orbit Module)

Accuracy
The accuracy calculation was reworked and is valid. The main issues are the dependence on a wavelength and the assumption of a configuration for the determination of the maximum baseline. An improvement could have been to calculate the maximum baseline with STK, but it would still have been dependent on the algorithm to organize the swarm and would have been time consuming computationally. The accuracy can be adapted if GPS is not used on the satellites and another system gives position and time information. Accuracy is important, but is linked with the ambiguity issue; a great accuracy is not worthwhile if the signal cannot be analyzed without any ambiguity. This places constraints on the geometric configuration of the swarm. (See the Time Module.)

Latency
Latency can be improved by implementing better models of on-board processing, communication delays in internal and external communications, and the impact of autonomy.

Revisit time (See Orbit Module)

Global coverage (See Orbit Module)

Mission completeness
Mission completeness is quite robust and accounts for the degradation of the system. Improvement in mission completeness will emerge from a better reliability model or realistically accounting for replenishment of satellites.

5.4.8.6 Verification

A Matlab test module was written simulating all the inputs needed by the module. This test code was useful to correct the syntax of the code. However, the main test was the first run. We

identified two main problems. The first one was incoherencies in the units of the attributes because of different units used in the orbit and utility codes. The second was major problems in the calculation of the AOA accuracy, mainly the problem of the instrument phase error that had been neglected and the value given to the instrument phase error in a second iteration.

5.4.9 Utility Module

5.4.9.1 Introduction

Fundamental to this module is the multi-attribute utility analysis (MAUA) taught in Dynamic Strategic Planning at MIT. (Please see Utility chapter for more information regarding MAUA.) This function takes in attribute values and, using the data from the utility interview in CONSTANTS, determines the single attribute utilities. It then uses the multi-attribute scaling factors in CONSTANTS to calculate the multi-attribute utility. The function loops this algorithm three times (for each time period: BOL, MOL, EOL). NOTE: the constant 3 should be renamed and moved to CONSTANTS since it appears in several modules. The utility code is contained within the *utility_function.m* and *calculate_K.m* files.

5.4.9.2 Required Inputs

The utility module takes inputs from the following modules:
CONSTANTS
ATTRIBUTES

The inputs are as follows:
CONSTANTS.utility.data_set_E,
CONSTANTS.utility.data_set_A:
These contain single attribute utility data from the utility interview, one for missions without the AOA mission (E) and one with (A).

CONSTANTS.utility.k_values_EDP,
CONSTANTS.utility.k_values_AOA:
These contain the multi-attribute scaling factors from the utility interview, one for missions without the AOA mission (EDP) and one with (AOA).

ATTRIBUTES.attribute_values:
This matrix has all of the attribute values in a row. Each row is a different time period. (e.g. BOL, MOL, EOL.) Comes from the calculate_attributes module.

5.4.9.3 Output Descriptions

UTILITY.single_attribute_util:
This matrix has all of the single attribute utilities in a row. Each row is a different time period. (e.g. BOL, MOL, EOL.)

UTILITY.multi_attribute_util:
This vector has as each element the multi-attribute utility at a different time period. (e.g. BOL, MOL, EOL.)

ERROR.utility_function.out_of_range:
Equals one if attribute is out of valid range of utility function, zero otherwise. Does not prevent

output of utility function, however. User must decide whether to use output utility. ERROR flag is output by output_btos module.

5.4.9.4 Key Assumptions

Fundamental equations

No fundamental physics is involved here. We use linear interpolation between data points to determine the single attribute utilities. A multiplicative multi-attribute utility equation is used to aggregate the single attribute values into a multi-attribute utility. (Please see Utility chapter for more detailed discussion of utility theory and process.)

Multiplicative equation:

$$KU(\underline{X})+1=\prod_{i=1}^{n=6}\left[Kk_iU(X_i)+1\right]$$

where:

- K is the solution to $K+1=\prod_{i=1}^{n=6}\left[Kk_i+1\right]$, and –1<K<1. This variable is calculated in the *calculate_K* function.
- $U(\underline{X})$, $U(X_i)$ are the multi-attribute and single attribute utility functions, respectively.
- n is the number of attributes (in this case six).
- k_i is the multi-attribute scaling factor from the utility interview.

Rationale for any simplifications

There are two key assumptions for use of this utility functional form.

- Preferential independence
- The preference of $(X_1',X_2') > (X_1'',X_2'')$ is independent of the level of X_3, X_4,…, X_n.
- Utility independence
- The "*shape" of the utility function* of a single attribute is the same, independent of the level of the other attributes. "Shape" means that the utility is the same up to a positive linear transformation, $U'(X_i)=aU(X_i)\pm b$.

Evolution of calculations

The function has basically remained the same since first written. The only changes involve the addition of a time period loop, error flag, and a rescaling of the AOA accuracy range in CONSTANTS.

5.4.9.5 Fidelity Assessment

Due to the nature of the interview, the utility values given by the customer are accurate to approximately ± 0.1 utility points. The measurement resolution of the single attribute utility function is to within 0.05 utility points. Performing sensitivity analysis to the function reveals that if all utility functions are off by 0.1 utility points, the multi-attribute result is off by about 0.004.

5.4.9.6 Verification

The code was verified by inserting extreme range values for the attributes to the utility function. End points (zero for all attributes at their worst, one for all attributes at their best, and in between for other combinations) were predictably output, both for the single attribute utilities and the multi-attribute utility. The *test_util*, *test_utility*, and *test_maua* functions were used for this verification.

5.4.10 Other Code

Other than the modules, B-TOS also has supporting code. These include the output routine contained within the *output_btos.m* file and the user interface code contained within the *BTOS.m* file. The code that calls all of the modules is contained within the *main.m* file. This main code is looped by the B-TOS shell code as specified by the user. Additional support code includes *num2cur.m*, which is a function that takes in a number and spits it out as a currency string, and *tradespace_enumerate.m*, which is a function that is run once to enumerate the tradespace of permutations of the design vector. (See Appendix on code usage for more information.) Along with the tradespace code is the *read_design.m* file that contains the code for translating the enumerated tradespace into a design vector for the B-TOS code. The environment code is contained within the *environment.m* file, however this module is not used in B-TOS. Lastly, *position.rst* is a support file for use with Satellite Tool Kit and is inherited from the A-TOS code.

5.5 Integration Process

Various modules in the simulation software were assigned to various sub-teams. The main issue the integration team faced was making sure the modules worked together. Because the class was only allowed two weeks to develop this software, the integration team realized that the integration issue must be addressed at the beginning of the development process to minimize rework at the end. The following actions were taken:

- Set variable and module conventions
- Develop I/O sheets
- Construct an N-squared Diagram

The rest of this section will discuss each action item in detail, and conclude with lessons learned.

5.5.1 Variable and module conventions

Since the code is developed using Matlab and Matlab is case-sensitive, the integration team required the module development teams to use consistent cases for the variables. The basic requirements are:

- Use lower case for variables in each module
- Use all capital letters for the output structures from each module

5.5.2 I/O sheets

The B-TOS architecture tradeoff software consisted of 11 main modules, not including many other sub-modules. The modules passed information between one another. The integration team needed to address the following issues:

1. Modules used the same names for the same variable.

2. The input variables that are needed by each module could get the necessary information from another module in the software.
3. The output variables produced in each module were needed by another module in the software.
4. The consistency and correctness of the input/output variables needed to be checked very frequently—at least once a day, or even once every hour during the final integration stage.

In order to achieve the above objectives, the integration team designed I/O sheets using Microsoft Excel. An example of the I/O sheets is in Figure 5-5. The features in the I/O sheets are designed to address the above four questions. They are explained in detail next.

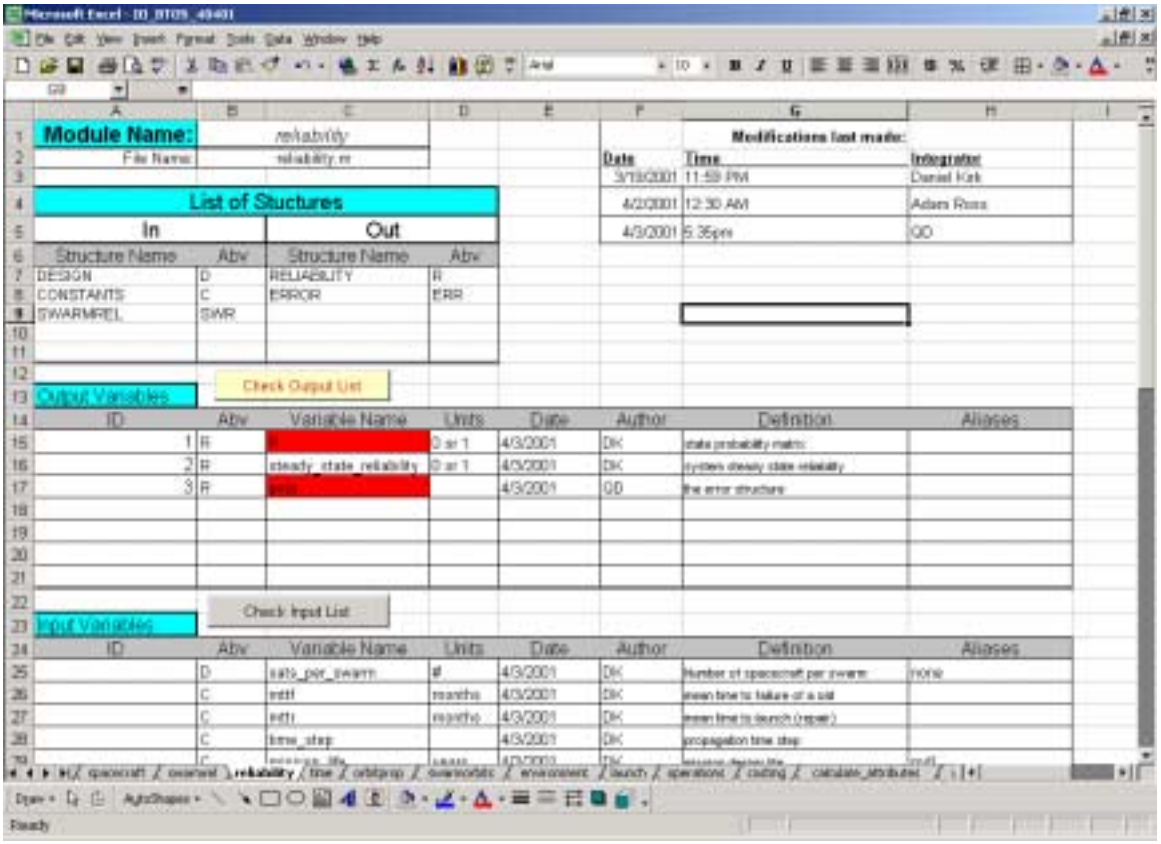


Figure 5-5 Example I/O Sheet

First, each module developer was asked to fill in their interface variables. The units and definition for each variable, as well as their names used in the program, are listed. This way, modules could verify consistency in their naming scheme and use the same variable names when needed. Explicitly listing the units prevented conversion errors and helped with code verification. In addition, the update time and author of the variables are listed so that if necessary, the corresponding person could be contacted.

Second, looking carefully on the sheets, one can see two buttons—"Check Output Variables" and "Check Input Variables". These two buttons are related to EXCEL Visual Basic codes embedded in the file. When the "Check Input Variable" button is clicked, all of the output variables in all of the sheets in this file will be searched, until an output variable that matches this input variable is found. If after searching all the sheets, no output variable matches the particular input variable, that variable will be flagged in red. The "Check Output Variable" button functions in a similar way to check if all of the output variables match an input variable in all the sheets in the file. These two buttons automated the interface variable checking procedure. In this way, the integration team was able to check the consistency of the interfaces among modules any time they wanted in a very efficient manner. When a red variable was discovered, the integration team contacted the responsible persons in various module teams involved and facilitated the management of the interfaces.

These I/O sheets helped a great deal in the final stage of the integration. The integrators were able to quickly see where the problems were at the interface, and fix the problems immediately. This would have been a very tough job if all of the variables at the interface had to be managed manually.

5.5.3 N-squared Diagram

An N-squared diagram was built in order to monitor the information flow among modules and facilitate the integration of modules. The N-squared diagram was initially constructed from the sequence diagram. Later on, it was updated based on the interaction provided in the I/O sheets. The final N-squared diagram is shown in Figure 5-6. The final relationship among the modules in the software is shown in Figure 5-7.

	Module Name	D	C	SW	SWR	R	O	ORP	L	OPS	Cost	T	A	SC	U	out
		Design	Constants	Swarm	Swarmrel	Reliability	Orbit	Orbitprop	Launch	Operations	Costing	Time	Calculate_Attribute	Spacecraft	Utility Function	output_BTOS
D	Design															
C	Constants															
SW	Swarm	x	x													
SWR	Swarmrel	x	x	x												
R	Reliability	x	x		x											
O	Orbit	x	x													
ORP	Orbitprop	x	x	x												
L	Launch	x	x	x												
OPS	Operations	x	x	x												
Cost	Costing	x	x	x					x	x						
T	Time	x	x		x											
A	Calculate_Attributes	x	x				x					x				
SC	Spacecraft	x	x													
U	Utility Function	x	x										x			
out	output_BTOS	x	x	x		x	x		x		x		x		x	

Figure 5-6 N-squared Diagram

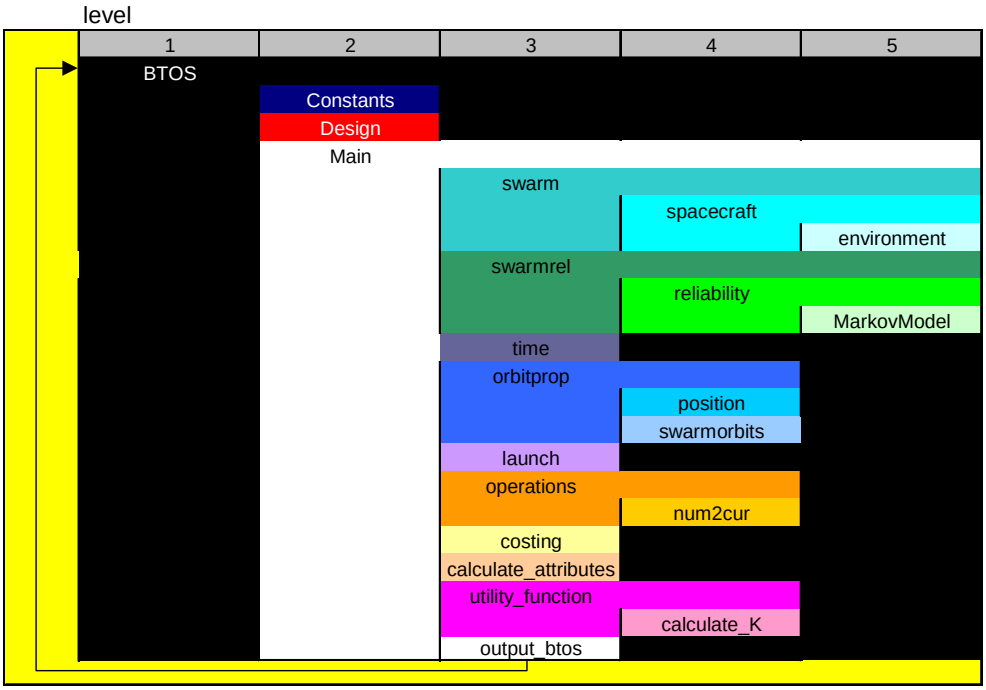


Figure 5-7 Module Information Flow Diagram

The N-squared diagram in Figure 5-6 shows that the interactions among the modules are linear and one directional. This observation matches with the initial design intention. The architecture of the software was intentionally designed for the data to flow in one direction, and the design achieved that goal.

During the process of the module development, the integrators updated the N-squared diagram regularly. Sometimes, the N-squared matrix revealed that iteration occurred between certain modules because of the I/O variables they listed. The integrators then called the related modules together and resolved the issue to eliminate the iterations if possible. In the end, all unnecessary iterations were eliminated and the software was designed as originally intended.

Another interesting observation is that the N-squared diagram shows that the design process of the software could have been a streamlined waterfall process. However, the actual software development process was highly iterative. Why is the reality so different from the final N-squared diagram? The software development process the team experienced was iterative because the class did not know what the exact interaction would be when the software development process started. The iterative process was the process to discover the interactions through trial-and-error. If future teams such as C-TOS were to develop a similar software program, they could start from this N-squared diagram and reduce many of the iterations in their development process.

5.5.4 Lessons Learned

Due to the time limitation, the integration phase of the development process was very challenging. The integration team found a few things that could be changed in order to make this work easier next time.

First, to manage the interface, the most important thing is to keep the I/O sheet of each module up to date. Due to the tight development schedule, filling in the I/O sheets were sometimes delayed. In the future, the teams should try to be more diligent with updating the I/O sheets daily.

Second, to reduce the integration work, each module should first verify their modules before bringing them to integration. Although the integration team had developed verification sheets for each module to fill out, due to time constraints, not all module verification sheets were properly filled in, and module level verification was not sufficiently done. Consequently, the amount of integration work at system level was increased.

Third, a positive learning experience came from the use of an error structure. Errors were not monitored in the A-TOS software module. In order to save computer-processing time and eliminate unreasonable results, B-TOS introduced the use of error variables and structures. When a module catches an error in its outputs or calculation, it raises a flag in the corresponding error variable. The Main module catches the error and acts accordingly. Most of the errors caused the program to terminate the consideration of its current design architecture and move on to the next one, with the exception of attribute errors. An attribute error usually occurs when the calculated attribute value is out of bounds. Sometimes the resulting attribute may actually be out of bounds on the good side—over-achieving our goal. In this case, the program flags the utility results and leaves the final judgment to the team.

Fourth, the timing of the school spring break was bad for our development efforts. Spring break caused a communication breakdown at a crucial time in the software development process. Most

people were away from campus, and it was hard to collaborate over emails. Next time, deadlines should be set either before breaks, or several weeks after.

In conclusion, the class as a whole learned a great deal from the integration process of this project. The learning and experience will benefit us in the real world.

6 Code Results

Since the designing and integration of the B-TOS code was iterative, there were several series of results. This section will only deal specifically with the results from the B-TOS Version 1.4 code. The design and integration teams made every effort to create a code that was as robust and as accurate as possible. Given the results, the code is quite capable of analyzing a multitude of architectures by varying the design variables (see Section 4.3) and outputting specific attributes that map to a corresponding utility value. This code is capable of varying orbital, swarm, and spacecraft parameters to measure relative architecture utility and cost. Given the high computational times associated with each architecture, it is critical to limit the number of architectures, thus limiting the tradespace enumeration to only those architectures that provide interesting and reasonable trades. After the enumeration and code run it is possible to compare different architectures with the first comparisons being based on the cost vs. utility plots. After recognizing a narrowed tradespace, greater detail about individual swarm performance can be gathered for frontier architecture analysis. In conducting this analysis it is important to consider the sensitivity of the model to variations in parameters that are known to have some level of uncertainty. Finally, these aggregate results shed light on future code modifications and more detailed studies.

6.1 Code Capability

The B-TOS code is currently capable of analyzing variable orbital geometries, multiple swarm size and density options, and spacecraft of individually varying functionality. Essentially, the code can take any combination of architectures specified by the design vector and output specific attributes that map to corresponding utility values.

It is important to understand that this code does not take input design vector and output an answer saying which architecture is the "best." Instead, the current model outputs a focused tradespace. It does not specify single-point architecture, but gives the cost and utility of each of the input architectures. The customer can then quickly look at a cost versus utility plot and see which of the possible architectures deserves further study.

Typically, the customer will be looking for the combination that gives the highest utility with minimal cost. He or she can look at the top left corner of this plot, pulling out likely architectures. Then the customer can further investigate each individual architecture's actual performance, as defined by the attributes the customer viewed as important.

While this model can be very effective in analyzing relative architectures, its true purpose must not be misunderstood or incorrectly applied. The model does not give "the answer," but this seems to be its strength because it directs the customer's attention to the most likely possibilities, making the first iteration of decisions based on function instead of design or requirements.

6.2 Tradespace Enumeration

One of the most critical aspects of making this code useful is generating a reasonable enumeration of the tradespace. Given that two of the design vector variables, altitude and swarm radius, are positive real numbers the tradespace could literally be infinite. Computation speeds are the limitation to broad tradespace analysis. In the enumeration outlined here, the run time

was approximately sixty five seconds per architecture. Obviously analyzing millions of architectures is out of the question, given most users' computational capacity. As a result, one must wisely choose the enumeration of the tradespace.

The B-TOS Architecture essentially has three levels within its tradespace: orbital, swarm, and spacecraft. An enumeration file was developed and input into the model. This file generated 4,033 different architectures, and required 73 hours of computation time on eight Pentium III processors.

6.2.1 Orbital Level Enumeration

The first part of the enumeration required making decisions regarding the likelihood of high utility values for the orbit and swarm variables. The table below shows the enumeration decisions for the orbital and swarm levels.

Table 6-1 Orbital and Swarm Level Enumeration Matrix

Design Vector Variable	Chosen Enumeration Values
Circular orbit altitude (km)	1100, 1300
Number of Planes	1, 2, 3, 4, 5
Number of Swarms/Plane	1, 2, 3, 4, 5
Number of Satellites/Swarm	4, 7, 10, 13
Radius of Swarm (km)	0.18, 1.5, 8.75, 50
5 Configuration Studies	Trades payload, communication, and processing capability

Above are the chosen design vector variable values for enumeration. Using this number of variable combinations gives a total of 4,033 architectures for analysis. Altitude was chosen based on Bill Borer's specification that top-side sounding could not be done below 1100 kilometers. One higher altitude was chosen to confirm the assumption that the model drives to the lowest possible altitude. The number of planes and swarms per plane were driven by an understanding that cost would become excessive for higher numbers of satellites. For instance, if there are 5 planes, 5 swarms/plane, and 13 satellites/swarm, the constellation would consist of 325 satellites, almost certainly cost prohibitive.

6.2.2 Swarm Level Enumeration and Swarm Geometry Considerations

Making prudent choices on the orbital radius proved to be one of the more complicated tasks of the enumeration. As shown in the above table, the selected radii are not completely intuitive. The selection process was iterative and driven by the maximum desired accuracy specified by the customer, which was 0.0005 degrees error of the angle of arrival determination. Recalling from

the attribute calculation module in section 5.4, the accuracy of the angle of arrival (dθ) was a function of the beacon wavelength (λ), the total phase error (dφ), and the baseline (D).

$$d\theta = \frac{\lambda}{2\pi D} d\varphi_{total}$$

Of these three parameters the first two are constants which are simply a function of the beacon hardware (λ = 3 meters, based on a transmit frequency of 100 MHz) on the ground, the sounding, and GPS hardware onboard the spacecraft (dφ = 1.099 radians, based on one nanosecond GPS time error, 10 centimeter GPS position error, and 15 degree sounding instrument error). The third, baseline, is a function of swarm radius, as indicated in the figure below.

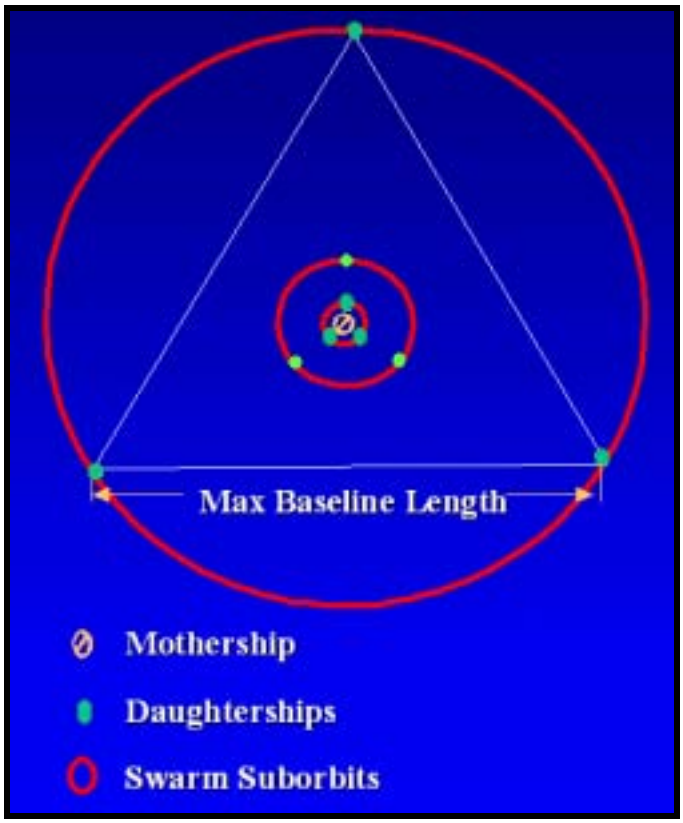


Figure 6-1 Swarm Geometry

The maximum baseline to achieve best accuracy was first determined to be approximately 86.6 km, corresponding to radius of about 50 km. This drove the selection of the outer-ring enumeration.

Figure 6-1 depicts a characteristic swarm geometry for ten spacecraft, with the mothership in the center and nine daughterships. In order to eliminate all ambiguity in the angle of arrival, the smaller baselines need to be filled. This fill is based upon a factor of 0.175, which is simply the

phase error ($d\varphi$ = 1.099 radians) divided by 2π. The inner radii are then selected by multiplying the outer radius by 0.175, hence the radii 1.5, 8.75, and 50 km. The inner radii must be filled until reaching the necessary minimum radius. The smallest baseline does not need to be any less than the wavelength (λ), divided by the accuracy of the onboard three orthogonal antennas. This accuracy is again a function of hardware. Given an accuracy of 0.017 radians and a frequency of 100 MHz, the minimum baseline does not need to be less than 176 meters. Again referring to the above table, instead of choosing 267 meters (the next radius after 1.5 km, based on the fill factor), 180 meters was used; however, the 267 meter radius would have been the more logical choice here. This was recognized after the code was run. The following results will show that this did not affect any of the key architecture trades.

Using this geometry, the number of satellites per swarm was given by number of satellites defining the triangle plus the center mothership. The number of triangles, or swarm sub-orbits is given by the number of fill radii discussed above. One other item to note regarding swarm geometry is the actual shape of the swarm. Currently, based on the explanations given by the aggregate customer, in order to make accurate angle of arrival determinations the baselines must be parallel to one another. Given that the above geometry should remain essentially constant relative to one another, the geometry should be maintained throughout the swarm propagation, meeting the required parallel orientation of baselines. Another factor driving geometry was the need to have baseline series that are non-parallel, simply one satellite needed to be non-collinear with the other two in order to make 3-D angle of arrival determinations. This implies a triangular configuration, and for reasons of orbital geometry an equilateral triangle seems most appropriate.

6.2.3 Enumeration for Configuration Studies

This third level of the design vector variables deals directly with the functionality of each individual spacecraft. While the code has the capacity to create a separate functionality combination for each spacecraft in the swarm, the enumerations for this run focused on functionalities of a mothership in the center of the swarm surrounded by "n" number of daughterships in the surrounding swarm sub-orbits. This enumeration considered five different functionality studies show in the figure below.

Table 6-2 Configuration Studies Matrix

Study	**1**		**2**		**3**		**4**		**5**	
Type	M	D	M	D	M	D	M	D	M	D
Number	**4+**	**0**	1	3+	1	3+	1	3+	1	3+
Payload (Tx)	Yes	n/a	Yes	Yes	Yes	Yes	**No**	Yes	**Yes**	**No**
Payload (Rx)	Yes	n/a	Yes	Yes	Yes	Yes	**No**	Yes	**Yes**	Yes
Processing	Yes	n/a	Yes	**No**	Yes	**Yes**	Yes	No	Yes	No
TDRSS Link	Yes	n/a	Yes	**No**	Yes	No	Yes	No	Yes	No
Intra-Swarm Link	**No**	n/a	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

In Table 6-2, there are five configuration studies listed with two different spacecraft types: a mothership (M) and a daughtership (D). The last four rows of the first column of the above figure lists the spacecraft-level design variables. The payload (Tx/Rx) refers to the capacity of the payload to sound (ping the ionosphere) or to receive the reflected signals off of the ionosphere. Spacecraft with processing are capable of compressing the data (assumed a 3:1 ratio). TDRSS Link is the spacecraft's long-range communication capacity to send information from the swarm to the surface via TDRSS. Finally, the intra-swarm link refers to the spacecraft's short-range communication systems, sending information to other spacecraft in the same swarm. The above studies have the following distinctions listed in Table 6-3; each study is listed with corresponding functionality differences between the mother and daughterships.

Table 6-3 Swarm Configuration Distinctions

Study 1	ALL **INDEPENDENT SPACECRAFT** THAT DO NOT COMMUNICATE WITH EACH OTHER, DOING LITTLE TO UTILIZE THE SWARM CONFIGURATION.
Study 2	INTRA-SWARM COMMUNICATION WITH ALL SPACECRAFT PINGING AND RECEIVING SIGNALS FROM THE IONOSPHERE, SENDING ALL INFORMATION TO THE **MOTHERSHIP FOR PROCESSING AND LONG-RANGE TRANSMISSION** TO TDRSS.
Study 3	INTRA-SWARM COMMUNICATION WITH ALL SPACECRAFT PINGING AND RECEIVING SIGNALS FROM THE IONOSPHERE, **INDIVIDUALLY PROCESSING** THAT INFORMATION THEN TRANSMITTING IT ALL TO THE MOTHERSHIP FOR LONG-RANGE TRANSMISSION TO TDRSS.
Study 4	INTRA-SWARM COMMUNICATION WITH **ONLY DAUGHTERSHIPS PINGING AND RECEIVING** SIGNALS FROM THE IONOSPHERE, SENDING ALL INFORMATION TO THE MOTHERSHIP FOR PROCESSING AND LONG-RANGE TRANSMISSION TO TDRSS.
Study 5	INTRA-SWARM COMMUNICATION WITH MOTHERSHIP PINGING AND RECEIVING SIGNALS FROM THE IONOSPHERE, AND **DAUGHTERSHIPS ONLY RECEIVING,** SENDING ALL INFORMATION TO THE MOTHERSHIP FOR PROCESSING AND LONG-RANGE TRANSMISSION TO TDRSS.

As will be indicated below, Study 5 yielded higher utilities than other configurations. This configuration calls for very simple daughterships with only the capacity to receive returns from the ionosphere, collecting that data and sending it without processing to the mothership.

After considering all of these possibilities as likely candidates for the final architecture, the code was enumerated and run to output 4,033 architectures. This data file was appended to the B-TOS Version 1.4 folder and the code was run, dividing up the different architectures between eight Pentium III computers.

6.3 Architecture Assessment and Comparison Methodology

B-TOS Version 1.4 was run, outputting to a data file: 1) run idea specifying version number, enumeration number, and computer; 2) all design vector variables; 3) average satellite mass and

power; 4) architecture total cost and error, and individual costs for spacecraft, operations, launch and IOC; 5) all attribute values and associated utility values.

While having output all of these series allows one to look at correlations between several of the parameters, the primary relationships of interests are the cost versus utility. Below is the entire enumeration plot. It is important to note that the x-axis is the *lifecycle cost.* This is the cost for the spacecraft, launch, and operations for five years. The five year lifecycle period was used for all output of B-TOS Version 1.4.

As indicated on the plot, the lower values are those architectures that were unable to conduct the beacon angle of arrival mission. Recall from section 4.3 that this was one of the design variables. Following plots will focus on the higher utilities. The second of the two focuses on those higher utilities, and also displays an interesting point regarding the swarm radii. In Figure 6-2, lifecycle cost vs. utility is plotted with utility ranges from 0.75 to 1.0 on the y-axis and logarithmically scaled lifecycle costs in millions of dollars on the x-axis.

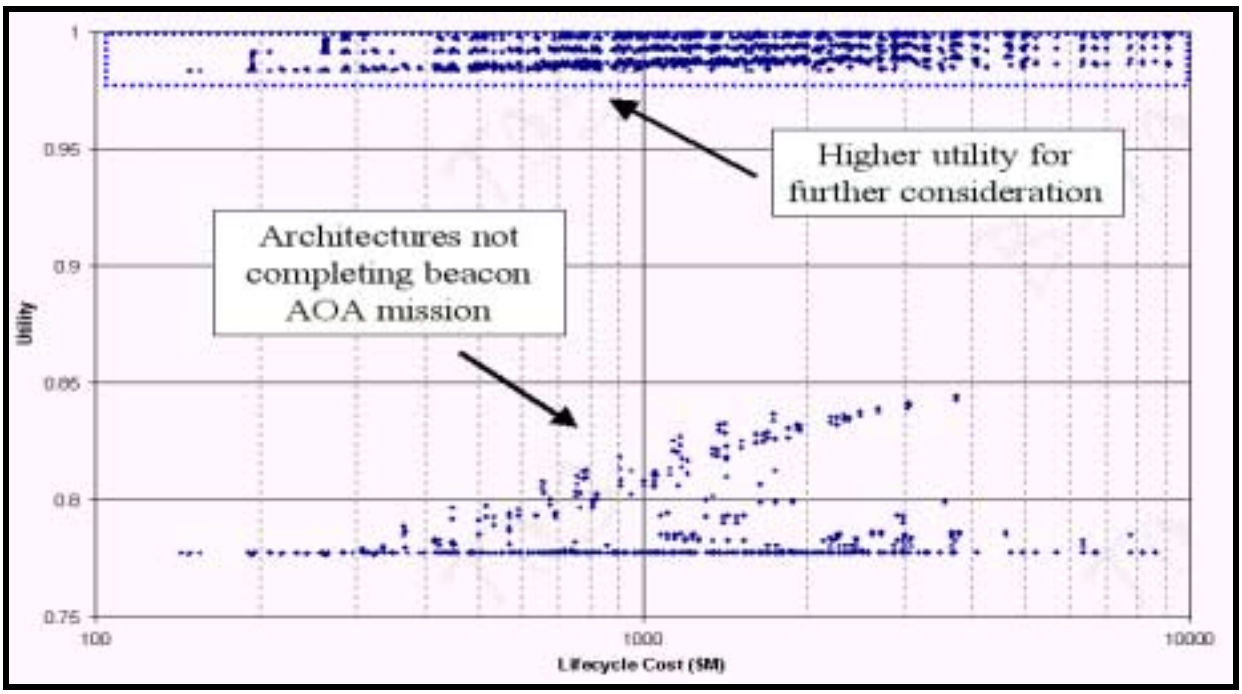


Figure 6-2 Cost vs. Utility for the Entire Enumeration

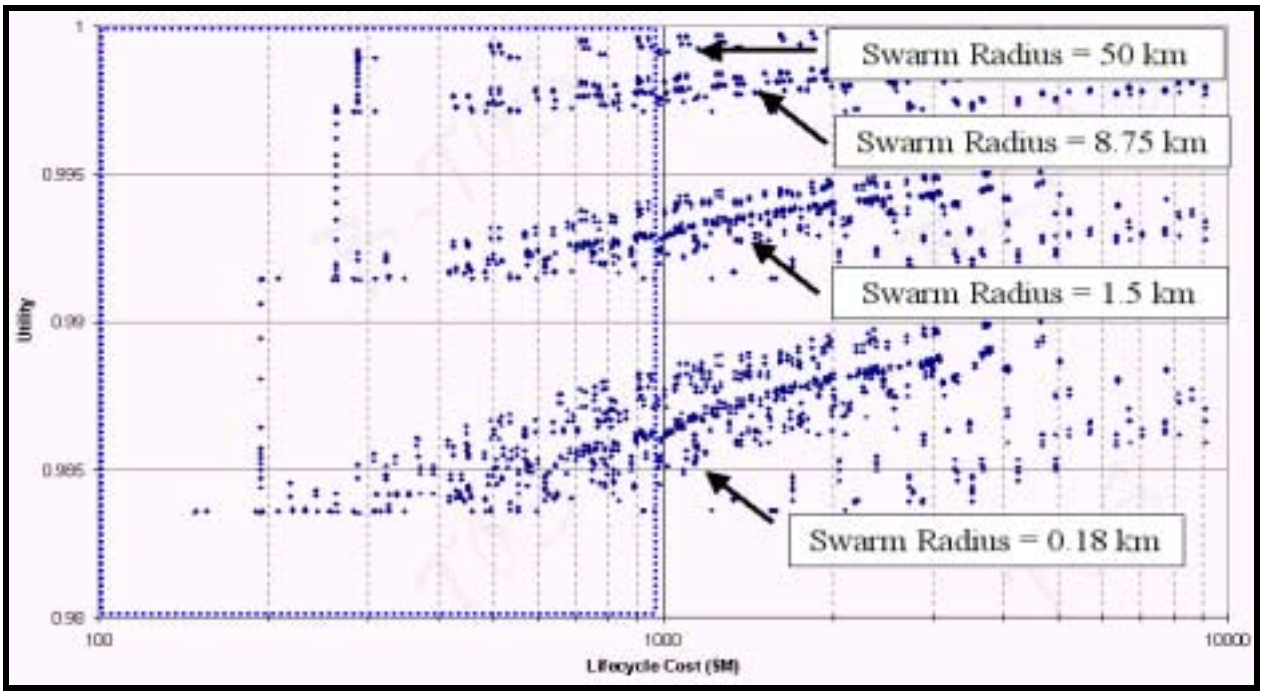


Figure 6-3 Cost vs. Utility (>.98) Swarm Radius

In Figure 6-3, notice the bands for each of the different swarm radii, increasing utility with increasing swarm radius. Note that this is only a subset of the whole enumeration. The above plot shows that as swarm radius increases the utility increases. This is primarily a result of the higher accuracies that come from the increased baseline length. Each band is correlated with the four different swarm radii selected for enumeration. One can recognize the difference in cost between the different radii looking for example the number of points less than one billion dollars for the 0.18 km band compared to the 50 km band at the top of the plot. In order to prevent ambiguity, more satellites are needed to fill as the swarm radius increases. This increase in number of satellites manifests itself in the increased cost.

The final cost vs. utility plot for analysis is shown below. This plot only considers those architectures with utilities greater than 0.98 and lifecycle costs less than one billion dollars. This plot highlights a few architectures of greatest interest.

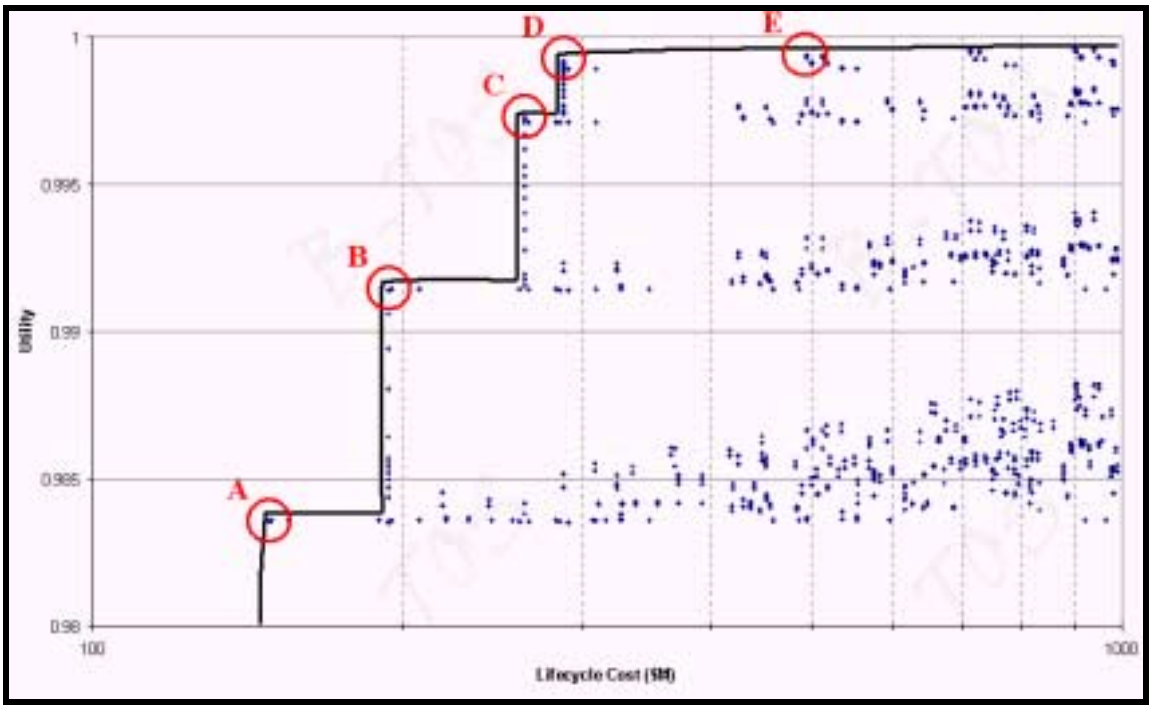


Figure 6-4 Cost (< $1B) vs. Utility (>.98) – The Knee

Figure 6-4 is again a plot of selected enumeration points. Additionally, the vertical lines highlight additional enumeration with the only change being swarm radius. Points A-E are considered the knee points that will be used for further analysis and indicate the relative lowest cost with highest utility. After the initial run of the code, another short enumeration was performed varying only swarm radius. These architectures are seen near the dark stepped line. This showed that the highest utility swarm was one that had the largest radius. Again, recognize that this model does not indicate the best architecture, but instead gives the customer a few key architectures on which to focus attention.

6.4 Frontier Architecture Analysis

In the previous figure architectures A, B, C, D, and E are identified. Returning to the data files, it is possible to reconsider the particular characteristics and the true attribute performance of each of these satellites. The following tables will elucidate some of the key differences between these different selected architectures.

Point	A	B	C	D	E
Altitude (km)	<– 1100 –>				
Num of Planes	<– 1 –>				
Swarms/Plane	1	1	1	1	2
Satellites/Swarm	4	7	10	13	13
Swarm Radius (km)	0.18	1.5	8.75	50	50
Functionality Study	<– #5 –>				

Recall:

Study	5	
Type	M	D
Number	1	3+
Payload (Tx)	Yes	No
Payload (Rx)	Yes	Yes
Processing	Yes	No
TDRSS Link	Yes	No
Intra-Swarm Link	Yes	Yes

Figure 6-5 Key Architecture Design Variables

In Figure 6-5, the top table shows the orbit and swarm level variables for architectures A-E. All five points turned out to be configuration study five, which is shown in the bottom table. The figure summarizes the design variables for the five different architectures. Notice that the main difference between the architectures is the different radii. Point E is an option with one more swarm per plane. Later, this will be indicated by an increase in re-visit time and increasing utility; however, the nominal increase in utility as indicated by the plot, comes at a significantly increased cost.

Returning to the output data allows a more detailed examination of the different architectures, specifically their performance seen in both the values for attributes and the total utility value. Additionally, the different costs are shown for both total lifecycle and IOC. The following plot can be presented to the customer for the customer to have a look at the most likely architectures from which to select. If there have been changes in customer preference since the utility interview the customer has the flexibility to choose the architecture based on adjusted preferences among the attributes, whose values are shown corresponding to each architecture.

Point	A	B	C	D	E
Spatial Resolution (deg)	4.36	5.25	7.34	9.44	9.44
Revisit Time (min)	805	708	508	352	195
Latency (min)	3.40	3.69	4.36	5.04	5.04
Accuracy (deg)	0.15	0.018	0.0031	0.00054	0.00054
Inst. Global Coverage	0.29%	0.29%	1.15%	2.28%	4.55%
Utility	0.9835	0.9914	0.9973	0.9992	0.9994
IOC Cost ($M)	90	119	174	191	347
Lifecycle Cost ($M)	148	194	263	287	494

Figure 6-6 Key Architecture Attributes, Utility, and Cost

For each of the specified points, the values for the five attributes are shown along with the associated utility value and IOC / Lifecycle costs in millions of dollars. Further detail may be considered for each of the architectures as well. For instance, the customer may want to get an idea of the spacecraft characteristics. Again, these data are part of the model output and can be relatively easily assembled for initial spacecraft design considerations. In this case, all architectures had spacecraft characteristics based on configuration study five and gave the below values. Additionally, cost can be analyzed for each different design point. Below is the cost distribution for "architecture C."

	Mothership	Daughter
Spacecraft mass (kg)	165	72
Subsystem mass breakdown:		
ADACS:	8	8
CDH:	10	4
Payload:	32	17
Power:	33	13
Propulsion:	11	11
Structures:	33	14
Telecom:	29	1
Thermal:	8	4
Downlink data rate (bps)	30000	15000
Average power required (W)	191	73

Figure 6-7 Spacecraft Characteristics

Figure 6-8 "Point C" Cost Distribution

Figure 6-7 gives estimated values for mass, data rates, and power for both the motherships and daughterships. Figure 6-8 shows the "Point C" cost distribution. The launch cost is for two Athena IIs. The total lifecycle cost is $ 263 million. All of the focused tradespace architectures have very high utilities. As will be discussed later in section 6.6 the magnitude of these values is not particularly important. The usefulness of these values comes in comparing relative values. While these utility values do not provide immediate conceptual descriptions, they can be correlated to attribute values. The customer can then compare relative architectures in the same terms in which he or she specified needs.

As indicated by Figure 6-5, competitive architectures must be able to conduct the beacon angle of arrival mission. This angle of arrival collection has an even greater value if the swarm is able to accurately characterize the angle of arrival with minimal error. This capacity comes through different swarm sizes. Figure 6-6 shows that the swarm radii are the key differentiators between architectures with high and low utility. In these enumerations, the highest utilities could only be achieved with the 50 km radius, 13-satellite architecture. Keep in mind that this does not mean that the best architecture will have these characteristics, especially when one considers the added cost of the additional satellites.

The most promising trades seem to be those with simple swarm geometry and single swarm missions. Again, the single swarm has fewer satellites and therefore a significantly lower cost. Furthermore, consolidated functionality on the mothership looks like the most likely candidate to properly achieve customer needs. This means that the mothership will be relatively complicated providing sounding capability, data processing, and long-range TDRSS transmission for the entire swarm. The daughterships, on the other hand, will be very simple, simply collecting the reflected signals from the surface of the ionosphere and from the beacons.

6.5 Sensitivity Analysis and Uncertainty

While the frontier architecture analysis is done based primarily on the cost vs. utility plots, it is important to understand that those points specifying a particular cost and utility are not completely accurate. Figure 6-9 shows the Multi-Attribute Utility Process. Notice that the outputs, cost and utility are on the far right and therefore, those values are only as accurate as their inputs. In order to better understand the accuracy, it is necessary to first look at some of the assumptions invoked at various places in the model. Then it is important to consider the accuracy of the utility function. Finally, with this background it is possible to do an analysis of the model, characterizing the models sensitivity to the assumptions, considering the ways in which variations in the constants space affects utility and cost.

Figure 6-9 MAUA Flow Chart

6.5.1 Assumptions

As with all models, B-TOS relies on a multitude of assumptions in order to make the problem tractable. First, there are some assumptions about the orbit. The swarms are assumed to be in a Walker Constellation. The model, using the Satellite Tool Kit (STK) performs a two-body propagation of the orbit and assumes that the orbit will have station-keeping capacity. Additionally, there is no calculation for the swarm maintenance delta-V. Instead, the model invokes an estimated delta-V per year. The orbit also assumes the ability to sustain swarm that is coplanar with uniform angular spacing between each satellite in the sub-orbit.

Based upon the early preferences of the aggregate customer, the design vector only provides mission options that include the EDP mission. In calculating the spatial resolution this model uses STK functions intended for optics mission and therefore spatial resolution is circular with an area defined by elapsed time between data set collection points. Along these same lines, there is no consideration of a field of view for the angle of arrival mission. The model assumes that one beacon is in view at all times.

Additionally, the customer stated that EDP missions were only possible above 1,100 km, and the model gives little value of higher altitudes. Altitude is considered in three places: 1) cost calculations, where lower is better; 2) in the velocity, which decreases with altitude, so spatial resolution gets better, creating nominal increases in utility; 3) conversely, the decreased velocity, increases revisit time, causing a decreased utility. Unless there is very high attribute value on spatial resolution, it will drive to the lowest altitude. This drive towards lower altitudes would be magnified if calculations were done showing the lower EDP accuracy and the higher payload power required from higher altitudes. The 1,100 km altitude does require some radiation hardening which is only accounted for in the cost module with a crude rule-of-thumb scaling factor for altitudes above 1,000 km. Other costing was done using a cost estimating relationship from SMAD (see section 5.4 for module description).

The communication and data-handling model also invokes several assumptions. One of the more glaring of these is the ability to use an infinite number of TDRSS links. Additionally, there is no communication delay between the satellites and no communication delay between the swarm and ground. Several assumptions were also made regarding the payload data rates and spacecrafts capacity for data processing.

Finally, while the model does perform reliability calculations, the reliability constants used in B-TOS version 1.4 run for a five year lifecycle and there was no difference between the beginning, middle, and end of life. Adding to this inadequacy is the models failure to calculate launch and deployment failures. Furthermore, there is no implementation for satellite replenishment, nor is there any consideration of launch scheduling. All launch vehicles are sized based on a satellite that is a square cylinder, with a volume based on an assumed density.

6.5.2 Utility Function Analysis

The two primary outputs of the B-TOS model for each architecture are cost and utility. The utility function used in B-TOS is described in detail in section 3.1. In order to prevent page turning, the function is re-written below.

$$KU(\underline{X})+1=\prod_{i=1}^{n=6}\left[Kk_iU_i(X_i)+1\right]$$

The functions $U_i(X_i)$ and the k_i values are derived from the utility interview. The K value is calculated from the k_i values.

The results for the architectures plot utility versus cost, with the top architectures differing in the third or fourth decimal place in utility. A reasonable question is whether there is any difference at all. What is the difference between 0.993 and 0.994? They both look good on a scale from 0 to 1. (Answer: a difference between 0.993 and 0.994 is a lot! But it also depends…)

The answer lies in the multiplicative nature of the function and the values of the k_is.
Example: For the interview conducted in this class, k_6=0.95. This represented the immense value the customer placed on having the B-TOS architecture perform both the AOA and EDP missions. If the utility from each of the other five attributes were all zero (at their worst acceptable level to the customer), the overall utility for the mission would be 0.95! (Quite misleading since five out of six attributes are at their worst level!) A mission performing well in all six attributes will accumulate a lot of 9s in this case. Thus, the difference between 0.993 and 0.994 may be performing well or not well in an attribute. In the end, the critical determination of the difference between two different utilities lies in converting the utility back to its attribute values. Once converted back to attribute-space, if there are noticeable differences in the attribute values, then the difference in utility is significant.

Now that the third or fourth decimal place of utility may be significant, the next reasonable question may involve uncertainty of the utility. Experimental uncertainty arises in the values of the single attribute utility functions and the k values. (Please see Appendix B for the questionnaires and data from the interviews.) The resolution of the single attribute utility function determination is +/-0.05. (The questioning procedure bracketed preferences down to this scale.) The resolution of the k values is +/-0.025 for the same reason. Sensitivity analysis was conducted on the utility function to assess the worst and average case errors in the multi-attribute utility if all of the utility answers were shifted by a resolution scale or two (i.e. if the "true" utility were +/-0.05 or +/-0.1 from the measured utility.) The same was conducted on the k values. The single attribute utilities were shifted by

$$U'(X_i) = U(X_i) + \delta,$$

where $U(X_i)$ is the single attribute utility assessed in the utility interview, δ (+/-0.05 or +/-0.1) is the shift value, and $U'(X_i)$ is the new single attribute utility value.

The k values were shifted by

$$k_i' = k_i + \varepsilon,$$

where k_i is the k_i value assessed in the utility interview, ε (+/-0.025 or +/-0.05) is the shift value, and k_i' is the new k_i value.

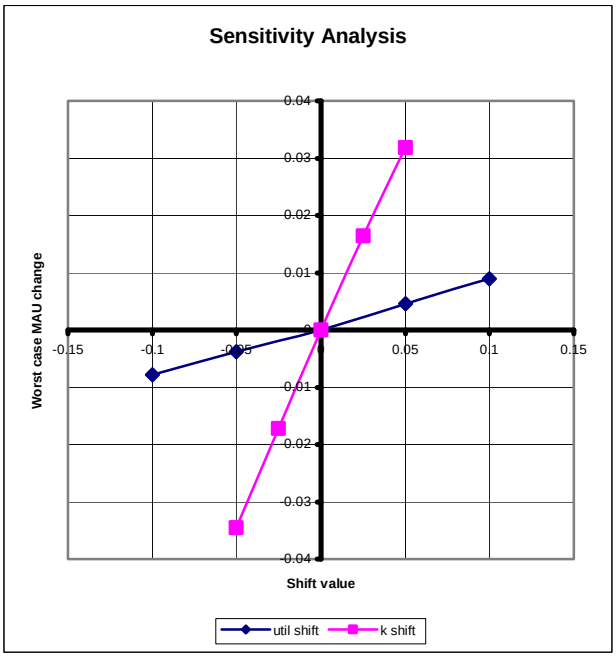


Figure 6-10 Worst Case MAU Plot

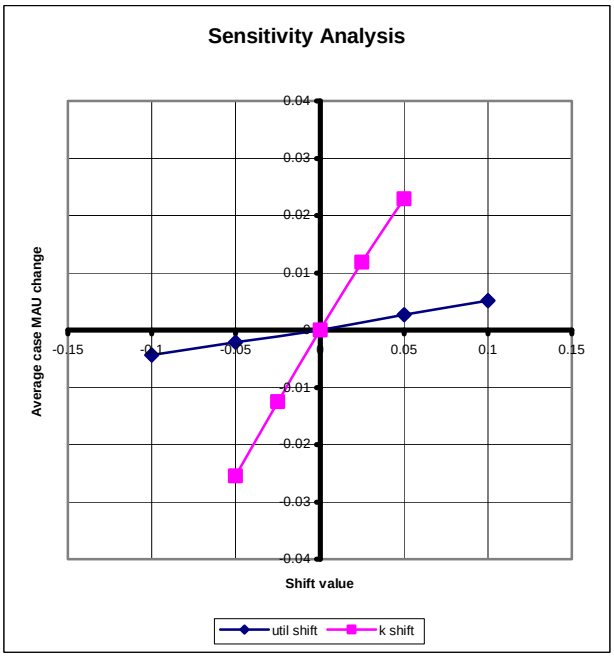


Figure 6-11 Average Case MAU Plot

Figure 6-10 and 6-11 show the error in multi-attribute utility (MAU) as a function of these shifts.

Constant linear shifts in all values were determined to be approximately the maximum error in the utility since on average the errors will not all be correlated (in the same direction) and thus the error would be less. Also, it is important to note that errors in *k* would have much more of an impact on the utility value. For this reason, in future interviews it is important to verify the *k* values and improve confidence in its value. It is also important to note that there may be no such thing as the "true" utility function for an individual since preferences are more of a fuzzy notion than a concrete one[8]. Also, preferences shift slightly from day to day. Thus there may be some inherent variance in the utility function and an "error" of a small shift in utility may still capture the essence of the customer's preference.

Another important issue to mention is inconsistencies between the original and validation interviews. (Please see utility section for more discussion of this issue.) Initial inconsistencies are a natural part of the utility interview process. The subject has a strong desire for self-consistency and will try to fix any inconsistency that crops up during the interview. It is part of the responsibility of the interviewer to point out inconsistencies and facilitate the subject in their rectification. The interviewer must be careful to not introduce bias into this process. In the case of the validation interview for B-TOS, the interviewer suffered from over cautiousness regarding bias and lack of experience spotting inconsistencies. This is a partial explanation of the inconsistencies between interviews. It is not believed that these inconsistencies represent

[8] Ralph L. Keeney and Howard Raiffa, Decisions with Multiple Objectives: Preferences and Value Tradeoffs, John Wiley & Sons, New York, NY (1976).

fundamental changes in the customer's preferences. Rather a manifestation of the lack of experience of the interviewers and the novelty of the process.

6.5.3 Model Analysis

Constants

The first step in analyzing the model was to consider which of the variables were not known with very high levels of certainty. Table 6-4 lists the constants that were recognized to have relatively high levels of uncertainty. Their values in B-TOS Version 1.4 are listed in the center column under the 0%. Each of the variables were adjusted by plus and minus five and ten percent. The only exceptions being the bottom shaded rows where the values were considered to have greater variability. The constants have their appropriate units listed except for the factors which for the time factors, for instance, were not specific times but instead represented a fraction of the orbital period where TDRSS was out of sight or when the spacecraft was conducting maintenance operations. All analysis was done based on "Point C."

Table 6-4 Sensitivity Enumeration Table

Constant Percent Change	-10%	-5%	0%	5%	10%
spacecraft mass factor	0.9	0.95	1	1.05	1.1
instrument phase error (deg)	13.5	14.25	15	15.75	16.5
beacon wavelength (Hz)	9.00E+07	9.50E+07	1.00E+08	1.05E+08	1.10E+08
gps time error (sec)	9E-10	9.5E-10	1E-09	1.05E-09	1.1E-09
gps position error (meters)	0.09	0.095	0.1	0.105	0.11
bearing (radians)	0.78534	0.82897	0.8726	0.91623	0.95986
flight software cost ($)	9.00E+06	9.50E+06	1.00E+07	1.05E+07	1.10E+07
edp time (secs)	35.1	37.05	39	40.95	42.9
beacon time (secs)	35.1	37.05	39	40.95	42.9
maintenance time factor	0.036	0.038	0.04	0.042	0.044
no tdrss time factor	0.036	0.038	0.04	0.042	0.044
ops scale factor	5.4	5.7	6	6.3	6.6
turb time (secs)	0	15	30	45	60
data set delay (secs)	0	15	30	45	60
mission life (years)	1	3	5	7	9

After recognizing these potentially inaccurate variables in the module, the code was run again using the updated-scaled values for each of the parameters. Thus, after doing this run it was possible to see which of the variables affected the utility output and which of the variables affected the cost output. As one might expect the following variables affected cost: 1) mission life; 2) spacecraft mass; 3) no TDRSS time; 4) operations scale factor. The rest of the variables

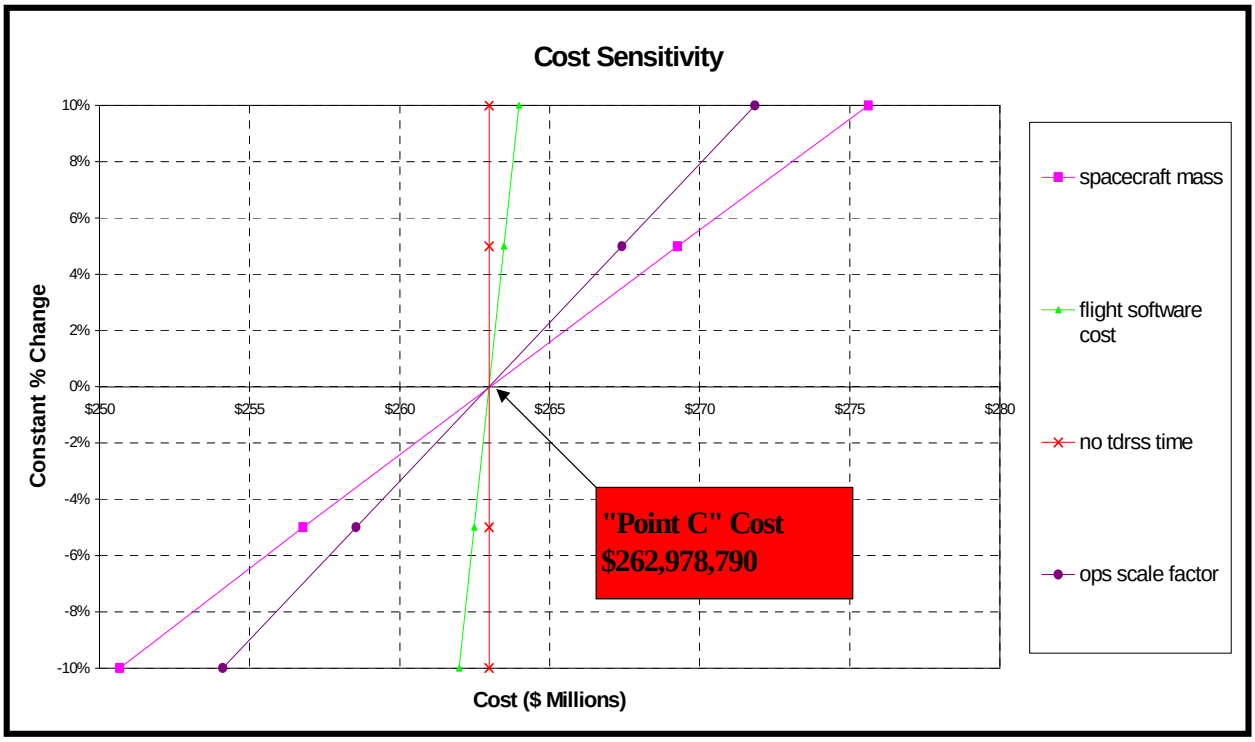


Figure 6-12 Cost Sensitivity

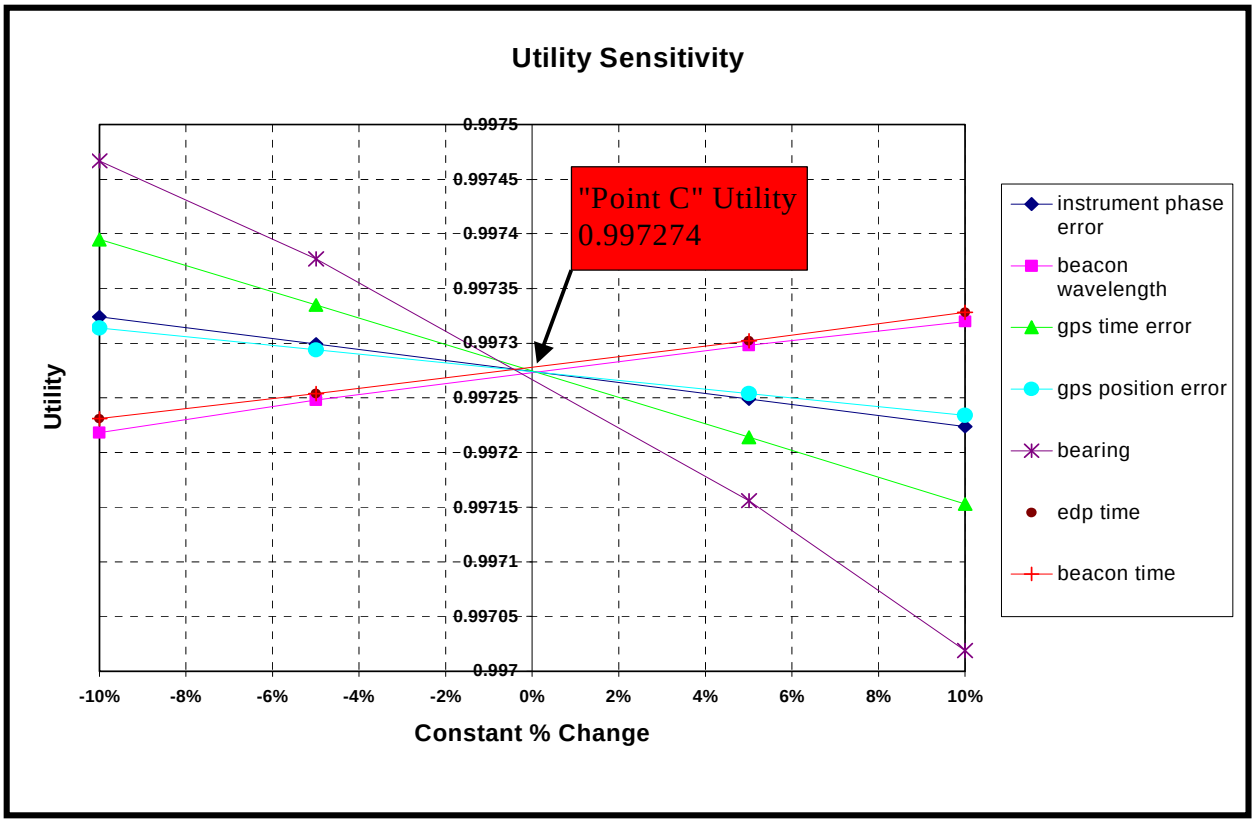


Figure 6-13 Utility Sensitivity

affected utility except for the maintenance time. This implies that either there is such a small effect, which means it had an affect of less than 0.000001 on the utility, or the code did not adequately account for this time. Figures 6-12 and 6-13 are the affects of constant changes on cost and utility.

The axes on both graphs are oriented in such a way that one can imagine the variability in the cost vs. utility plots previously, with cost being on the x-axis and utility being on the y-axis. These plots show the possible inaccuracies in the points shown in Figures 6-2, 6-3, and 6-4. In some ways, these sensitivity graphs could be considered error bars on the previous cost vs. utility plots. The accuracy to which cost and utility are known depend on the accuracy of these constants and the accuracy of the model in converting these constants through a physical system into accurate attribute outputs that can be converted into a utility value.

Figure 6-12 shows how the changes in the constants affect the cost of the architecture. Again, these costs are total lifecycle costs. In this graph the more vertical the line, the less sensitive the model is to the given variable (look for movement left and right similar to the left and right placement of cost on the cost vs. utility graphs). As previously stated, TDRSS time was one variable affecting cost; however, as indicated by the nearly vertical line this effect is nominal. Additionally, as one would expect, the cost is most sensitive to the spacecraft mass with an essentially linear relationship for this region, with 10% errors in mass resulting in approximately 5% (almost $13 million in this case) errors in cost.

Figure 6-13 has more variables to consider. Note that on this graph, the more horizontal the lines the less sensitive the model is to the given variable (this time look for movement up and down the axis similar to the up and down utility in the cost vs. utility graphs). Notice that some slopes are positive and some are negative. This simply means that increasing error could either increase or decrease the utility, depending on which constant it is. This is expected. If just one constant is off by 10% it could change the utility by as much as .00027. Remember from Figure 6-6 that the utility difference between architecture D and E was .0002. Again, recall from 6.5.2 that utility magnitudes are not completely meaningful. Percent changes here will not provide the same intuitive sense as percent changes in the cost.

The bearing angle shows some degree of non-linearity. Looking at the accuracy calculations in the module descriptions one finds that this constant appears in the numerators and denominators of all of the terms of the accuracy calculation has a sine or cosine operation performed on it. Furthermore, with what has been said about utility and the importance of the angle of arrival attribute it is understandable that the model would be sensitive to this constant. In some ways this is problematic in determining the expected accuracy. It is important to understand that the model has angle of arrival as constant in order to compare the different architectures, but in actuality this value changes continuously as the swarm propagates around the earth receiving from one beacon and then from the next.

Mean Time To Failure (MTTF)

One capability of the code that was not used in the B-TOS Version 1.4 run is the ability to determine utility at the beginning, middle, and end of life. As previously stated, for this particular run there was no difference between the three periods. This leads one to question the MTTF thresholds necessary to see a change in utility. At a short enough MTTF one of the components will fail, causing loss of one functionality and losing the capacity to perform to all

of the attributes. Below is a plot of utilities for given MTTFs for the three different periods, beginning (BOF), middle (MOF), and end of life (EOF).

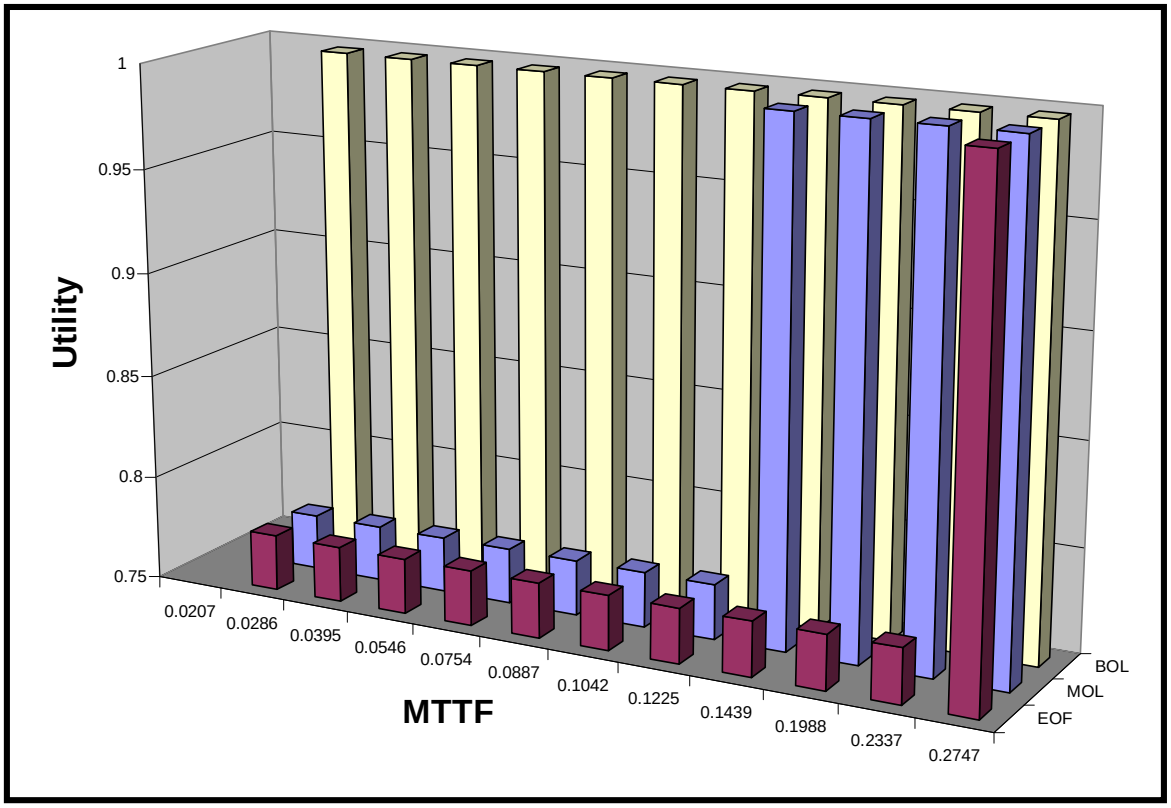


Figure 6-14 Mean Time To Failure

Notice that the different MTTFs at different periods output one of two utilities, 0.776379 or 0.997273. Essentially, there was a failure of one of the spacecraft that prevented it from doing the angle of arrival mission, dramatically reducing the utility of the architecture. As indicated, if accurate MTTFs are entered into the models, constant vector the model should provide a relatively good approximation of the affect of accuracy on the utility.

6.6 Future Code Modifications and Studies

While the code currently seems to provide relatively accurate comparisons of architectures there are several areas that could be improved upon. These improvements include both greater research in determining accurate constants and more working in ensuring that all factors in the modules are accurately calculated by considering more variables. Given the discussion of the key trades between architectures swarm geometry, payload data handling, reliability and beacon angle of arrival, all need further consideration to improve upon this model.

6.6.1 Swarm Geometry

One facet of the code that certainly needs further study is the orbital geometry and the implementation of that geometry. As stated above, the assumed geometry is relatively simple and was chosen without in-depth consideration of competing geometries. Several of the assumptions leading to the selected geometry should be more rigorously confirmed. The delta-V required to maintain the assumed orbit is also a question. Also, it is not completely clear the level of position error, i.e. the variability of baseline lengths that is allowed to maintain the zero ambiguity.

An analysis by Hugh McManus showed that the swarm design, as described earlier in this document, could have problems with orbital disturbances. The J2 effect on the mothership and the daughters is different enough to create a strong tendency for the daughters to leave the formation. Countering this effect can create unrealistically large delta-V requirements for the outer-most daughters, especially in the larger swarms (e.g. frontier designs "D" and "E"). This effect is most pronounced if the mother and daughter vehicles have orbits with different inclinations or eccentricities. Swarm designs are, however, available with large radii and relatively low delta-V requirements, but they are not the simple Hill's constellations used here. The outermost vehicles may need to be in a leader-follower position, or on orbits with differing ascending nodes but identical inclinations and eccentricities. The difficulty with these lower-delta-V swarm designs is that their relative positions, and hence coverage patterns and accuracy, are more complex functions of time than the simple Hill's swarms.

This problem is confounded with other problems in the modeling of the swarm geometry. As noted above, the coverage area is only approximated, and is not a reflection of the true geometry of the swarm, even for the simple Hill's swarms used. Large, low delta-V swarm designs are available, but would have complex coverage patterns and accuracy that would be shifting functions of time. Choosing between them, or trading their virtues for the penalties in delta-V (but possibly also rewards in coverage) that come with the Hill's swarms, would require modeling of the coverage patterns and accuracy as a 3-dimensional and dynamic functions of time. The coverage patterns could then be used in somewhat more sophisticated utility functions, and the delta-V calculations used in more complete cost functions, to evaluate the tradeoffs in swarm geometry.

Given the availability of the low-delta-V swarm geometry, it is reasonable to say that the analyses presented here are not invalidated by the problems above, but a level of unmodeled complexity is clearly present. The performance (in terms of coverage) and cost (in terms of delta-V-requirement) are in fact more complex function of swarm geometry than was modeled. However, their is no reason to suspect that the more advanced models would show different trends than the very simple models used in the initial architecture study - e.g. larger swarms will have better AOA accuracy and coverage, with a cost penalty driven by the number of vehicles required.

6.6.2 Payload Data Handling

Current data rates listed in the code are far from accurate. These data rates are just constants in the constants module. The assumptions for the Version 1.4 run were on the order of 10 kbps. Since that time information has come from the aggregate user suggesting data rates of each

satellite would be on the order of 1-2 Mbps. This certainly changes the accuracy of the latency calculations and the number of TDRSS links.

In addition to the data rate constants, more modifications need to be made to account for the processing compression ratios. The specific type of processor was not considered and the compression ratio was a very rough approximation. Understanding more about the specific type and form of data being collected is critical to creating better data handling approximations. Furthermore, more must be understood about the types of intra-swarm communications systems, and the policy and technical limitations of using TDRSS links.

6.6.3 Reliability

While the model does perform reliability calculations, the reliability constants used in B-TOS version 1.4 run for a five-year lifecycle there was no difference between the beginning, middle, and end of life. The payload reliability is completely uncertain. Further studies need to be done to characterize that reliability so that it can be entered into the model. Also, there are reliabilities for know combinations of sub-systems that should be employed.

6.6.4 Beacon Angle of Arrival

It would be important to determine the maximum angles of arrival that can be detected by the system. As shown in the 6.5.3 sensitivity analysis, the high angles of arrival lead to low utilities as a result of low accuracy of this attribute. Also, it is assumed that one beacon is always in view of the swarm. Some simulation could be done, placing the beacons at their actual locations and determining how much of the time they really are in view given the maximum intelligible angles of arrival. Finally, the beacon frequencies affect the radius of the swarm (see section 6.2.2). Again, this was recognized as one of the important variables to trade in the design vector.

6.7 Summary of Key Results and Recommendation

Essentially, after running the code five key architectures were identified. All of them very closely meet the needs of the customer with slight differences in attributes that the customer can examine and decide upon an architecture with the most preferred attributes. To develop more accurate trade model there are several areas requiring further research. Overall, for the first round of a conceptual architecture this model is quite useful.

7 Conclusions

7.1 Process Summary

In completing this project, the following process was performed. First, the value propositions from the professors, customers, and students were collected to determine what each group wanted from this project. Next, a mission statement was written in order to provide a general statement of purpose and to help focus the team.

The utility function was developed by first identifying and creating a list of system attributes. The attributes are parameters that describe the quality of a system architecture. Interviews with the customer were completed to discover the customer's relative importance of each of the attributes. This was then translated into a mathematical utility function that could translate architectures' attributes into a utility ranging from 0 (worst) to 1 (best).

A list of design variables (also called the design vector) was then created. The design vector consisted of input variables to the computational model of the system. The values of the variables in the design vector would be allowed to vary to create different system architectures. A Quality Function Deployment (QFD) was used to map the design vector to the system attributes and to eliminate extraneous variables to make the design vector a manageable size. The design space was then defined by determining appropriate ranges for design vector variables using physical and system constraints.

The computational model of the system was developed by partitioning the problem into modules that calculated system attributes based on design vector inputs. Teams were created to develop each module, or set of modules. The modules were written primarily by modifying the code and structure created by A-TOS. An integration team was also created to keep track of inputs and outputs, make sure that teams communicated, and assemble all the modules into a fully working model of the system.

The model was then used to evaluate all possible meaningful architectures with respect to the utility function. This was accomplished by using the model to iterate across the design space, thereby creating thousands of unique satellite system architectures. The values for utility and other attributes could then be used to compare the thousands of architectures. For example, comparing utility and cost allows one to focus only on those architectures that are economically feasible. The customers can then choose the best architecture(s) that fit their needs. One particular architecture was selected and a rough first order design of the 'Mothership' was created.

7.2 Accomplishments

Throughout the course of this project, the class had some important accomplishments. Utility analysis was completed to capture the quality of system architectures, providing the ability to trade thousands of different designs. This allows system engineers to look at a broad spectrum of designs and choose a design that best fits their needs. To facilitate future analysis and direction, the tradespace has been narrowed to those architectures that are most feasible and provide the best utility for cost.

A detailed computational model of the system was created using Matlab. The code is robust, modular, and easy to upgrade. It can accommodate distinct satellite types with different functionality combinations.

7.3 Lessons Learned

Many lessons were learned throughout the process of completing this project. The most important lesson is that consistent and clear communication within the team, faculty, and customers is indispensable to the success of the project. Communication within the team and faculty was facilitated by three weekly meetings, web-based file sharing tools (Docushare), and emails. However, it was hindered by a lack of consistent vocabulary and evolving definitions of variables. Often different teams would have different ideas on the definition of a variable or process, which led to confusion and hindered the integration of the software code. Spring break also added difficulty to communication at a crucial time for the project. Constant communication with the customer was also critical, especially since this was a learning process for both the team and the customer. Communication with the customer provided direction and continually guided the progress of the project.

There were also many lessons learned during the process of separating and integrating the code amongst the team. The use of an N^2 diagram helped to determine the input/output (I/O) relationships between the different modules of the code. The diagram shows how to arrange the modules in order to create a 'waterfall' process, where modules are called in a linear fashion, simplifying the I/O structure. The N^2 diagram is good at capturing stable processes and improving them. It was also found that the process of learning about the relationships between modules is highly iterative. When trying to integrate the modules, it was found to be very important to accurately and routinely update I/O sheets for each of the modules. In addition, having individual module verification reduced the workload on the integration team. A standardized method of error trapping was also found to be useful, but should have been implemented at the beginning of the code development.

7.4 Results Summary

After running the code and producing thousands of different system architectures, the results were examined and some important trends were discovered and conclusions were made. The results show that architectures must collect beacon angle of arrival data to be in the higher utility segment of the tradespace. Among these architectures, swarm radii becomes a key differentiator. Larger swarm radii tend to produce greater utility. However, it was also found that larger swarm radii put greater demands on formation keeping and dramatically increase the required fuel loads, especially on the outer satellites.

The most promising and feasible system architectures tend to revolve around simple systems. These systems often have simple orbital geometries, consist of a single swarm, and consolidate functionality on the mothership with less functionality on the daughters.

8 References

Fowler, M. and Scott, K., *UML Distilled, second edition*. Addison-Wesley: Boston, 2000.

Keeney, Ralph L. and Raiffa, Howard, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, John Wiley & Sons: New York, NY, 1976.

Kong, E.M.C., D. W. Miller, and R. J. Sedwick, "Exploiting Orbital Dynamics for Aperture Synthesis Using Distributed Satellite Systems: Applications to a Visible Earth Imager System", *Journal of Astronautical Sciences*, Vol. 47, Nos. 1-2, Jan-Jun 1999.

Larson, Wiley J. and Wertz, James R., *Space Mission Analysis and Design 3rd ed.* Microcosm Press: El Segundo, CA., 1999.

Neufville, Richard. *Applied Systems Analysis: Engineering Planning and Technology Management*, McGraw-Hill Publishing Co.: New York, NY 1990.

Sabol, Chris, et al. "Satellite Formation Flying Design and Evolution." Journal of Spacecraft and Rockets, Vol. 38, No. 2, March-April 2001.

Sedwick, R.J., D.W. Miller aand E.M.C. Kong, "Mitigation of Differential Perturbations in Clusters of Formation Flying Satellites", *Journal of Astronautical Sciences*, Vol. 47, Nos. 3-4, Jul-Dec 1999.

Sediwick, R.J., E.M.C. Kong and D.W. Miller, "*Exploiting Orbital Dynamics and Micropropulsion for Aperture Synthesis Using Distributed Satellite Systems: Applications to TechSat 21*", AIAA-98-5289, AIAA Defense and Civil Space Program Conference, Huntsville, AL, October 28-30, 1998.

Shaw, Graeme B. *The Generalized Information Network Analysis Methodology for Distributed Satellite Systems*, MIT Thesis Aero Sc. D., 1999.

T. Tascione, *Introduction to the Space Environment*. Krieger Publishing: Malabar, FL 1994.

The DocuShare web site has a number of documents in the Papers and Reports folder. The Ionosphere and Instrumentation and the Related Projects subfolders have the most relevant background information.

SSPARC DocuShare:
http://leanair4.mit.edu/docushare/default.htm
including A-TOS architecture code

Background Information on Ionosphere and Topside Sounding
Ionosphere:
http://www.windows.ucar.edu/spaceweather/sun_earth9.html

Scintillation:
http://www.nwra-az.com/ionoscint/sp_news.html

National Space Weather Program:
http://www.ofcm.gov/nswp-ip/tableofcontents.htm

Space Weather Architecture Study:
http://www.acq.osd.mil/nssa/majoreff/swx/swx.htm

Ionosphere Topside Sounding:
http://www-ssc.igpp.ucla.edu/IASTP/93/
http://www.centerforremotesensing.com/Projects/sounder.htm
http://charley.izmiran.rssi.ru/~pulse/topside.htm (original reference)
http://193.232.24.27/projects/IK19/texts/articles.html (original reference)
http://www.kurasc.kyoto-u.ac.jp/~epic/MIR-sounder.html
http://www.ee.surrey.ac.uk/EE/CSER/UOSAT/IJSSE/issue1/palmer/palmer.html

Ground-based Ionosphere Sounding:
http://www.ngdc.noaa.gov/stp/IONO/ionogram.html

Related Programs
ION-F:
http://www.nanosat.usu.edu/overview/ion-f.html (original reference)

TechSat21:
http://www.vs.afrl.af.mil/TechProgs/TechSat21/

C/NOFS:
http://www.te.plk.af.mil/contracts/cnofs/cnofs.html (original reference)
http://www-vsbi.plh.af.mil/cnofs.htm (original reference)

SCINDA:
http://www-vsbi.plh.af.mil/scinda.htm (original reference)

ISIS & Alouette:
http://nssdc.gsfc.nasa.gov/space/isis.html (original reference)

Intercosmos 19:
http://193.232.24.27/projects/IK19/texts/ik-19.html (original reference)

Appendix A

Code Use Instruction

A1 Overview

The code interface and usage is mostly inherited from the interface of A-TOS. It evolved from a testing version and gained added functionality based on the needs of other programmers. Parallel processing application of this code also resulted in additional functionality for the user. A GUI could and should be developed for future versions of the code (perhaps C-TOS or later). The code itself is written in Matlab. In order to run B-TOS, the user needs all of the code files (25 of them), Matlab, and Satellite Tool Kit (STK) with applicable licenses. At this stage, the user also needs some experience with Matlab in order to define the inputs to the code (i.e. enumerate the tradespace).

A2 Necessary Files

In order for B-TOS version 1.4 to run, the user must have **Matlab 5.x** or higher and **STK 4.1.1b** or higher. The STK licenses *Mexconnect* (Matlab) and *Connect* must also be installed.

The following files need to be located in the same directory on the computer:

1. BTOS.m
2. calculate_attributes.m
3. calculate_K.m
4. constants.m
5. costing.m
6. design.m
7. environment.m
8. last.dat
9. launch.m
10. Main.m
11. MarkovModel.m
12. num2cur.m
13. operations.m
14. orbitprop.m
15. output_btos.m
16. position.rst
17. read_design.m
18. reliability.m
19. spacecraft.m
20. swarm.m
21. swarmorbits.m
22. swarmrel.m
23. time.m
24. tradespace_enumerate.m
25. utility_function.m

A3 Preparation

Before executing the code, it is necessary for some basic familiarity with the code. It is assumed that the user will be somewhat familiar with the code structure (as per section 5 of this document). The constants.m file contains all of the constants used by the code, including interface toggles. At the top of the constants.m file are the toggles most likely to be changed by the user. Here the user can change the naming convention for the output files from the B-TOS code. (Comments within constants.m refer to these fields.)

The first step before any execution of the code is to decide upon a tradespace enumeration. If no tradespace is enumerated, the code will look to the Design.m file for the design architecture to evaluate. (This feature had some bugs and it is unclear if they were resolved.) Edit the tradespace_enumerate.m file to decide the portion of the tradespace to be explicitly enumerated. (Typically only design variables are varied over some range and exhaustively listed in a very large matrix.) The tradespace_enumerate.m program will write a file called tradespace_btos.mat. This file, once generated, will allow the user to search part or all of the enumerated space. It only has to be generated once.

To generate the tradespace, after editing the tradespace_enumerate.m file, open Matlab and set path to the directory containing all of the B-TOS files. Also set the working (current) directory to the same directory. Type:

```
> tradespace_enumerate
```

The code will appear to pause as it enumerates. This only has to be done once, so it should not add significant time to the total run time. The code will tell the user when it has completed.

Now it is time to run the B-TOS code. Decide beforehand which part of the tradespace the computer should examine. (The tradespace_enumerate function creates a matrix containing *N* design vectors, where *N* is the total number of permutations coded in tradespace_enumerate.m) For example, if *N*=3500, the area of the tradespace that may be interesting to the user could be 2200-2850. In this case, the starting point would be 2200 and the number of iterations would be 650. Also the tradespace could be divided up equally in order to parallel process on multiple computers.

Open the constants.m file and change the CONSTANTS.initials value to a unique identifier for the computer/run. (Note: the results are time stamped, so it is possible to back out the information if the files are named the same, though this is not recommended.) Additionally, make sure the CONSTANTS.ouput_to_file_flag is set correctly. Use “log” ouput when investigating many architectures in a single run. Use “file” output when investigating single or few architectures in depth. The “log” output writes a single line of data per architecture and appends each new architecture to the same file, whereas the “file” output generates a detailed report per architecture.

Before running the code, be sure to start STK and close any open dialog boxes.

To run the B-TOS code, at the prompt type:

```
»BTOS
```

The following output will display on the screen:

```
--------------------------------
| Welcome to BTOS version 1.4 |
--------------------------------


Setting Constants...

Please enter tradespace enumeration number to begin:
```

Now enter the number of the tradespace enumeration for the first architecture input to B-TOS. An error message will let the user know if the number exceeds the tradespace size. If no number is entered, by default the code will continue from the last architecture investigated. (The last.dat file is created by the code and contains the architecture number of the last investigated architecture. If this file does not exist, the default is one.) The code then outputs:

```
Please enter number of loops to perform:
```

Now enter the number of iterations for the code to investigate. B-TOS version 1.4 moves linearly through the tradespace, incrementing the current tradespace by one in each loop. The code will terminate after the last architecture is investigated. If no number is entered, by default the code will perform one loop and then terminate.

B-TOS version 1.4 has a new feature that allows the user to enter the initials of the computer/run at this time. The next code output is:

```
Please enter computer intials ($$):
```

(Spelling error needs to be corrected…) Now enter the initials for the computer/run. If no initials are entered, the default value is defined in constants.m as CONSTANTS.initials (see above). This feature facilitates the parallel computing process whereby multiple runs are simultaneously started with each computer having different initials and different starting points in the enumerated tradespace.

The code now executes with varying screen output depending on toggle flags in constants.m.
Sample screen output:

```
Reading Design parameters...
Using architecture iteration #500
Evaluating Swarm Module...
Evaluating Reliability Module...
Evaluating Time Module...
Evaluating Orbit Module...
Warning: mexConnect:  Connecting to localhost:5001

Evaluating Launch and Deploy Module...
Evaluating Operations Module...
Evaluating Costing Module...
Evaluating Attribute Calculation Module...
Evaluating Utility Function...
With AOA mission

Finished evaluating Design#500

Finished running BTOS model.
```

After the code has finished execution, the output file(s) should appear in the current (working) directory. The file will end in *.gin*.

Appendix B

B-TOS Multi-attribute Utility Interview

The utility interview went through two iterations. They will be discussed separately in this section.

B1 Initial Multi-attribute Utility Interview (3.21.01)

Attributes	**Value Range**
1. Spatial Resolution	(1x1-50x50)
2. Revisit time	(5 minutes-720 minutes)
3. Latency	(15 minutes-120 minutes)
4. Accuracy	EDP: (100%-70%), AOA: (0.005 deg - 0.5 deg)
5. Instant Global Coverage	(100%-5%)
6. Mission Completeness	(1/2/3 - 1)

LEP: $(X^*, P_i, X_*) \sim (X_i, 0.5, X_*)$

Ask question by plugging in the first attribute value in the listed sequence and move through the suggested probability sequence (nested loop). Bracket probabilities until indifferent.

B.1.1 Example Questions

Example to familiarize customer with question format:

0. Price of car ($) (range: $1000 - $25000)

Your car has been giving you problems and you realize that you'll need to find a replacement soon. After long consultation with yourself, you decide that there are two options: buy a used car, or a new one. A used car will cost less in the short run, but has a risk that it will require more money to maintain it in the long run. A new car will cost more in the short run, but is less likely to require more infusions of money, however it could be a lemon and drop dead right away. Your town has only one dealership, so you can't shop around, however you do have a consumer guide that gives you the probability of failure for cars.

You have studied the consumer guide and it indicates that a new car will give you a 50% chance of costing you **XX** or $25000. A used car will give you a **##** chance of costing $1000 or a **1-##** chance of costing $25000. Do you go with the new or used car?
##: (Probability sequence: 45%, 10%, 35%, 20%, 25%)
XX: (Price sequence: $15000, $20000, $7000)

U($1K)=1
U($25K)=0

Single Attribute Function Questions:

1. Spatial Resolution (SR)

A research team is developing a new top-side sounder technology. It has not yet been demonstrated but has the possibility of increasing spatial resolution performance compared to the currently available instrument. You are at the stage in your design process where you have to decide which technology to implement.

Your design team has studied the issue. They indicate that the current technology will give you a 50% chance of getting a spatial resolution of **XX** or 50x50 deg. The new technology will give you a **##** chance of getting a spatial resolution of 1x1 degree or a **1-##** chance of getting 50x50 degree spatial resolution. Which technology would you choose?
##: (Probability sequence: 45%, 10%, 35%, 20%, 25%)
XX: (Spatial Resolution sequence: 25x25, 40x40, 5x5); (10x10)

U(50x50)=0
U(1x1)=1

2. Revisit time (RT)

Revisit time is solely a function of onboard processing capability. Your software team has developed a new plug-in for your currently available software. As a non-demonstrated technology, the new plug-in may increase or decrease the performance of the system, which will directly influence your revisit time capability. You are at the point in your design process where you have to choose whether or not to implement the new plug-in.

Your software team has studied the issue. They indicate that the current software will give you a 50% chance of getting a revisit time of **XX** or 12 hours. The new plug-in will give you a **##** chance of getting a revisit time of 5 minutes or a **1-##** chance of getting a revisit time of 12 hours. Do you choose to implement the new plug-in?

##: (Probability sequence: 45%, 10%, 35%, 20%, 25%)
XX: (Revisit time sequence: 1 hour, 30 minutes, 4 hours, 10 minutes)

U(5 minutes)=1
U(12 hours)=0

3. Latency (L)

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 communication system will give you a 50% chance of getting a latency value of **XX** or 2 hours. The new satellite communication system will give you a **##** chance of getting a latency value of 15 minutes or a **1-##** chance of getting a latency value of 2 hours. Which communication system would you use?

##: (Probability sequence: 45%, 10%, 35%, 20%, 25%)
XX: (Latency sequence: 40 minutes, 25 minutes, 1 hour); (90 minutes)

U(15 minutes)=1
U(2 hours)=0

4. Accuracy (A) (2 accuracy questions were asked, one for AOA and one for EDP)

A research team is developing a new top-side sounder technology. It has not yet been demonstrated but has the possibility of increasing accuracy performance compared to current available instrument. You are at the stage in your design process where you have to decide which technology to implement.

Your design team has studied the issue. They indicate that the current technology will give you a 50% chance of getting an accuracy of **XX** or 70%. The new technology will give you a **##** chance of getting an accuracy of 100% or a **1-##** chance of getting 70% accuracy. Which technology would you choose?

##: (Probability sequence: 45%, 10%, 35%, 20%, 25%)
XX: (Accuracy sequence: 90%, 95%, 80%); (85%)

U(100%)=1
U(70%)=0

5. Instantaneous Global Coverage (IGC)

Instantaneous global coverage is solely a function of the number of satellites, which is solely a function of budget. You have two options for funding. You can take the government's offer, which is option 1. Or you can apply for funding from a rich guy in the Cayman Islands, who is currently in Las Vegas gambling the money.

Suppose with option 1 you have a 50% chance to get an instantaneous global coverage of **XX** or 5%. Suppose with option 2 you have a **##%** chance to get instantaneous coverage of 100% and **1-##**% of getting 5%. Which funding would you choose?

##: (Probability sequence: 45%, 10%, 35%, 20%, 25%)
XX: (Instant Global Coverage sequence: 50%, 35%, 75%, 15%)

U(100%)=1
U(5%)=0

6. Mission Completeness (MC)

Mission completeness is solely a function of the number of different types of measurements you are able to take (i.e. AOA, EDP, and turbulence). These measurements are taken by separate instruments, which are currently supplied by one supplier. This supplier foresees the possibility of a strike, which may preclude them from supplying your instrument needs. Your other option is to get each instrument from a different supplier, which means that you may only end up with either 1 or 2 of the 3 instruments.

Suppose with option 1 you have a 50% chance to get **XX** measurements or just an EDP measurement. Suppose with option 2 you have a **##%** chance to get EDP, AOA, and turbulence and **1-##**% of getting just an EDP measurement. Which supplier scheme would you choose?

##: (Probability sequence: 45%, 10%, 35%, 20%, 25%)
XX: (Mission Completeness: EDP and AOA, EDP and Turbulence)

U(EDP, AOA, and Turbulence)=1
U(EDP)=0

B.1.2 Multi-attribute Function Questions (for corner points)

Variables: (SR, RT, L, A, IGC, MC)

Ka~ @(1x1, 12 hours, 2 hours, 70%, 5%, EDP)
Kb~ @(50x50, 5 minutes, 2 hours, 70%, 5%, EDP)
Kc~ @(50x50, 12 hours, 15 minutes, 70%, 5%, EDP)
Kd~ @(50x50, 12 hours, 2 hours, 100%, 5%, EDP)
Ke~ @(50x50, 12 hours, 2 hours, 70%, 100%, EDP)
Kf~ @(50x50, 12 hours, 2 hours, 0.5deg, 5%, EDP/AOA/Turbulence)

Ka: You can choose between having (1x1, 12 hours, 2 hours, 70%, 5%, EDP) for sure, or a ## chance of getting (1x1, 5 minutes, 15 minutes, 0.005 deg, 100%, EDP/AOA/Turbulence) and a **1-##** chance of getting (50x50, 12 hours, 2 hours, 70%, 5%, EDP). At what probability for the lottery would you be indifferent?

Kb: You can choose between having (50x50, 5 minutes, 2 hours, 70%, 5%, EDP) for sure, or a ## chance of getting (1x1, 5 minutes, 15 minutes, 0.005 deg, 100%, EDP/AOA/Turbulence) and a **1-##** chance of getting (50x50, 12 hours, 2 hours, 70%, 5%, EDP). At what probability for the lottery would you be indifferent?

Kc: You can choose between having (50x50, 12 hours, 15 minutes, 70%, 5%, EDP) for sure, or a ## chance of getting (1x1, 5 minutes, 15 minutes, 0.005 deg, 100%, EDP/AOA/Turbulence) and a **1-##** chance of getting (50x50, 12 hours, 2 hours, 70%, 5%, EDP). At what probability for the lottery would you be indifferent?

Kd: You can choose between having (50x50, 12 hours, 2 hours, 100%, 5%, EDP) for sure, or a **##** chance of getting (1x1, 5 minutes, 15 minutes, 0.005 deg, 100%, EDP/AOA/Turbulence) and a **1-##** chance of getting (50x50, 12 hours, 2 hours, 70%, 5%, EDP). At what probability for the lottery would you be indifferent?

Ke: You can choose between having (50x50, 12 hours, 2 hours, 70%, 100%, EDP) for sure, or a **##** chance of getting (1x1, 5 minutes, 15 minutes, 0.005 deg, 100%, EDP/AOA/Turbulence) and a **1-##** chance of getting (50x50, 12 hours, 2 hours, 70%, 5%, EDP). At what probability for the lottery would you be indifferent?

Kf: You can choose between having (50x50, 12 hours, 2 hours, 0.5 deg, 5%, EDP/AOA/Turbulence) for sure, or a **##** chance of getting (1x1, 5 minutes, 15 minutes, 0.005 deg, 100%, EDP/AOA/Turbulence) and a **1-##** chance of getting (50x50, 12 hours, 2 hours, 70%, 5%, EDP). At what probability for the lottery would you be indifferent?

##: (Probability sequence: 45%, 10%, 35%, 20%, 25%)

B.1.3 Initial Interview Results

Attribute	Value	Indifference Point	Utility
Spatial Res.	25x25 deg	0.325	**0.65**
	40x40 deg	0.05	**0.1**
	5x5 deg	0.49	**0.98**
	10x 10 deg	0.425	**0.85**
Revisit Time	60 min.	0.425	**0.85**
	30 min.	0.475	**0.95**
	240 min.	0.225	**0.45**
	540 min.	0.05	**0.1**
	40 min.	0.375	**0.75**
	15 min.	0.475	**0.95**
	60 min.	0.225	**0.45**
	90 min.	0.125	**0.25**
Accuracy (AOA)	0.16 deg.	0.175	**0.35**
	0.04 deg.	0.225	**0.45**
	0.01 deg.	0.425	**0.85**
	0.36 deg.	0.125	**0.25**
Accuracy (EDP)	90%	0.425	**0.85**
	95%	0.475	**0.95**
	80%	0.225	**0.45**
	85%	0.375	**0.75**
Inst. Global Cov.	50%	0.48	**0.96**
	35%	0.425	**0.85**
	10%	0.175	**0.35**
	15%	0.3	**0.6**
Mission Completeness	EDP and Turb	0.075	**0.15**
	EDP and AOA	0.475	**0.95**

Multi-attribute Corner Points

Attribute	k-value
Spatial Resolution	**0.15**
Revisit Time	**0.35**
Latency	**0.4**
Accuracy	**0.9**
Instant Global Coverage	**0.05**
Mission Completeness	**0.95**

B2 B-TOS MAUA Validation Interview Questionnaire (4.02.01)

Attributes	**Value Range**
6. Spatial Resolution	(1x1-50x50)
7. Revisit Time	(5 minutes-720 minutes)
8. Latency	(1 minute-120 minutes)
9. Accuracy	EDP: (100%-70%), AOA: (0.005 deg- 0.5 deg)
10. Instant Global Coverage	(100%-5%)
6. Mission Completeness	(1/2/3 - 1)

Lottery Equivalent Probability: $(X^*, P_i, X_*) \sim (X_i, 0.5, X_*)$

Ask question by plugging in the first attribute value in the listed sequence and move through the suggested probability sequence (nested loop). Bracket probabilities until indifferent.

B.2.1 Sample Questions

Two types of questions are used. The first type is the utility independence questions, and the second type is a set of mixed questions.

B.2.1.1 Utility Independence Questions

1. Spatial Resolution (SR)

A research team is developing a new top-side sounder technology. It has not yet been demonstrated but has the possibility of increasing spatial resolution performance compared to the currently available instrument. You are at the stage in your design process where you have to decide which technology to implement.

Your design team has studied the issue. They indicate that both technologies give you a revisit time of 5 minutes, a latency of 1 minute, an accuracy of 0.005 mrad, a global coverage of 100% and a complete mission (EDP, AOA, Turbulence). They indicate also that the current technology will give you a 50% chance of getting a spatial resolution of 25x25deg or 50x50 deg. The new technology will give you a **##** chance of getting a spatial resolution of 1x1 degree or a **1-##** chance of getting 50x50 degree spatial resolution. Which technology would you choose?
##: (Probability sequence: 45%, 10%, 35%, 20%, 25%)

2.Revisit Time (RT)

Time resolution is solely a function of onboard processing capability. Your software team has developed a new plug-in for your currently available software. As a non-demonstrated technology, the new plug-in may increase or decrease the performance of the system, which will directly influence your time resolution capability. You are at the point in your design process where you have to choose whether or not to implement the new plug-in.

Your software team has studied the issue. They indicate that both solutions give you a spatial resolution of 1x1 deg, a latency of 1 minute, an accuracy of 0.005 mrad, a global coverage of 100% and a complete mission (EDP, AOA, Turbulence). They indicate also that the current software will give you a 50% chance of getting a time resolution of 1 hour or 12 hours. The new plug-in will give you a **##** chance of getting a time resolution of 5 minutes or a **1-##** chance of getting a time resolution of 12 hours. Do you choose to implement the new plug-in?
##: (Probability sequence: 45%, 10%, 35%, 20%, 25%)

3. Latency (L)

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 both systems give you a spatial resolution of 1x1 deg, a revisit time of 5 minutes, an accuracy of 0.005 mrad, a global coverage of 100% and a complete mission (EDP, AOA, Turbulence). They indicate also that the current satellite communication system will give you a 50% chance of getting a latency value of 40 minutes or 2 hours. The new satellite communication system will give you a **##** chance of getting a latency value of 15 minutes or a **1-##** chance of getting a latency value of 2 hours. Which communication system would you use?

##: (Probability sequence: 45%, 10%, 35%, 20%, 25%)

4. Accuracy (A)

A research team is developing a new top-side sounder technology. It has not yet been demonstrated but has the possibility of increasing accuracy performance compared to current available instrument. You are at the stage in your design process where you have to decide which technology to implement.

Your design team has studied the issue. They indicate that both technologies give you a spatial resolution of 1x1 deg, a revisit time of 5 minutes, a latency of 1 minute, a global coverage of 100% and a complete mission (EDP, AOA, Turbulence). They indicate also that the current technology will give you a 50% chance of getting an accuracy of 1 mrad or 10 mrad. The new technology will give you a **##** chance of getting an accuracy of 0.005 mrad or a **1-##** chance of getting 10 mrad accuracy. Which technology would you choose?

##: (Probability sequence: 45%, 10%, 35%, 20%, 25%)

5. Instantaneous Global Coverage (IGC)

Instantaneous global coverage is solely a function of the number of satellites, which is solely a function of budget. You have two options for funding. You can take the government's offer, which is option 1. Or you can apply for funding from a rich guy in the Cayman Islands, who is currently in Las Vegas gambling the money.

Suppose both options give you a spatial resolution of 1x1 deg, a revisit time of 5 minutes, a latency of 1 minute, an accuracy of 0.005 mrad and a complete mission (EDP, AOA, Turbulence). Suppose with option 1 you have a 50% chance to get an instantaneous global coverage of 50% or 5%. Suppose with option 2 you have a **##%** chance to get instantaneous coverage of 100% and **1-##**% of getting 5%. Which funding would you choose?

##: (Probability sequence: 45%, 10%, 35%, 20%, 25%)

6. Mission Completeness (MC)

Mission completeness is solely a function of the number of different types of measurements you are able to take (i.e. AOA, EDP, and turbulence). These measurements are taken by separate instruments, which are currently supplied by one supplier. This supplier foresees the possibility of a strike, which may preclude them from supplying your instrument needs. Your other option is to get each instrument from a different supplier, which means that you may only end up with either 1 or 2 of the 3 instruments.

Suppose with both options you have a spatial resolution of 1x1 deg, a revisit time of 5 minutes, a latency of 1 minute, an accuracy of 0.005 mrad and a global coverage of 100%. Suppose with option 1 you have a 50% chance to get EDP and AOA measurements or just an EDP measurement. Suppose with option 2 you have a **##%** chance to get EDP, AOA, and turbulence and **1-##**% of getting just an EDP measurement. Which supplier scheme would you choose?

##: (Probability sequence: 45%, 10%, 35%, 20%, 25%)

7. Spatial Resolution (SR)

A research team is developing a new top-side sounder technology. It has not yet been demonstrated but has the possibility of increasing spatial resolution performance compared to the currently available instrument. You are at the stage in your design process where you have to decide which technology to implement.

Your design team has studied the issue. They indicate that both technologies give you a revisit time of 12 hours, a latency of 2 hours, an accuracy of 70%, a global coverage of 5% and only EDP measurement. They indicate also that the current technology will give you a 50% chance of getting a spatial resolution of 25x25deg or 50x50 deg. The new technology will give you a **##** chance of getting a spatial resolution of 1x1 degree or a **1-##** chance of getting 50x50 degree spatial resolution. Which technology would you choose?
##: (Probability sequence: 45%, 10%, 35%, 20%, 25%)

8.Revisit Time (RT)

Time resolution is solely a function of onboard processing capability. Your software team has developed a new plug-in for your currently available software. As a non-demonstrated technology, the new plug-in may increase or decrease the performance of the system, which will directly influence your time resolution capability. You are at the point in your design process where you have to choose whether or not to implement the new plug-in.

Your software team has studied the issue. They indicate that both solutions give you a spatial resolution of 50x50 deg, a latency of 12 hours, an accuracy of 70%, a global coverage of 5% and only EDP measurement. They indicate also that the current software will give you a 50% chance of getting a time resolution of 1 hour or 12 hours. The new plug-in will give you a **##** chance of getting a time resolution of 5 minutes or a **1-##** chance of getting a time resolution of 12 hours. Do you choose to implement the new plug-in?

##: (Probability sequence: 45%, 10%, 35%, 20%, 25%)

9. Latency (L)

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 both systems give you a spatial resolution of 50x50 deg, a revisit time of 12 hours, an accuracy of 70%, a global coverage of 5% and only EDP measurement. They indicate also that the current satellite communication system will give you a 50% chance of getting a latency value of 40 minutes or 2 hours. The new satellite communication system will give you a **##** chance of getting a latency value of 15

minutes or a **1-##** chance of getting a latency value of 2 hours. Which communication system would you use?

##: (Probability sequence: 45%, 10%, 35%, 20%, 25%)

10. Accuracy (A)

A research team is developing a new top-side sounder technology. It has not yet been demonstrated but has the possibility of increasing accuracy performance compared to current available instrument. You are at the stage in your design process where you have to decide which technology to implement.

Your design team has studied the issue. They indicate that both technologies give you a spatial resolution of 50x50 deg, a revisit time of 12 hours, a latency of 2 hours, a global coverage of 5% and only EDP measurement. They indicate also that the current technology will give you a 50% chance of getting an accuracy of 90% or 70%. The new technology will give you a **##** chance of getting an accuracy of 100% or a **1-##** chance of getting 70% accuracy. Which technology would you choose?

##: (Probability sequence: 45%, 10%, 35%, 20%, 25%)

11. Instantaneous Global Coverage (IGC)

Instantaneous global coverage is solely a function of the number of satellites, which is solely a function of budget. You have two options for funding. You can take the government's offer, which is option 1. Or you can apply for funding from a rich guy in the Cayman Islands, who is currently in Las Vegas gambling the money.

Suppose both options give you a spatial resolution of 50x50 deg, a revisit time of 12 hours, a latency of 2 hours, an accuracy of 70% and only EDP measurement. Suppose with option 1 you have a 50% chance to get an instantaneous global coverage of 50% or 5%. Suppose with option 2 you have a **##%** chance to get instantaneous coverage of 100% and **1-##**% of getting 5%. Which funding would you choose?

##: (Probability sequence: 45%, 10%, 35%, 20%, 25%)

12. Mission Completeness (MC)

Mission completeness is solely a function of the number of different types of measurements you are able to take (i.e. AOA, EDP, and Turbulence). These measurements are taken by separate instruments, which are currently supplied by one supplier. This supplier foresees the possibility of a strike, which may preclude them from supplying your instrument needs. Your other option is to get each instrument from a different supplier, which means that you may only end up with either 1 or 2 of the 3 instruments.

Suppose with both options you have a spatial resolution of 50x50 deg, a revisit time of 12 hours, a latency of 2 hours, an accuracy of 10 mrad and a global coverage of 5%. Suppose with option 1 you have a 50% chance to get EDP and AOA measurements or just an EDP measurement. Suppose with option 2 you have a **##%** chance to get EDP, AOA, and turbulence and **1-##**% of getting just an EDP measurement. Which supplier scheme would you choose?

##: (Probability sequence: 45%, 10%, 35%, 20%, 25%)

B.2.1.2 Random Mix Questions

Variables: (SR, RT, L, A, IGC, MC)

a~ @(25x25, 5 minutes, 60 minutes, 80%, 45%, EDP)
b~ @(50x50, 2 hours, 5 minutes, 90%, 30%, EDP)
c~ @(5x5, 30 minutes, 15 minutes, 0.005 deg, 55%, EDP/AOA/Turbulence)
d~ @(30x30, 4 hours, 1 hour, 0.25 deg, 30%, EDP/AOA)
e~ @(10x10, 6 hours, 20 minutes, 75%, 95%, EDP)
f~ @(20x20, 40 min, 30 min, 0.5 deg, 60%, EDP/AOA/Turbulence)

a: You can choose between having (25x25, 5 minutes, 60 minutes, 80%, 45%, EDP) for sure, or a ## chance of getting (1x1, 5 minutes, 1 minute, 100%, 100%, EDP) and a **1-##** chance of getting (50x50, 12 hours, 2 hours, 70%, 5%, EDP). At what probability for the lottery would you be indifferent?

b: You can choose between having (50x50, 2 hours, 5 minutes, 90%, 30%, EDP) for sure, or a **##** chance of getting (1x1, 5 minutes, 1 minute, 100%, 100%, EDP) and a **1-##** chance of getting (50x50, 12 hours, 2 hours, 70%, 5%, EDP). At what probability for the lottery would you be indifferent?

c: You can choose between having (5x5, 30 minutes, 15 minutes, 0.005 deg, 55%, EDP/AOA/Turbulence) for sure, or a **##** chance of getting (1x1, 5 minutes, 1 minute, 0.005 deg, 100%, EDP/AOA/Turbulence) and a **1-##** chance of getting (50x50, 12 hours, 2 hours, 0.5 deg 5%, EDP). At what probability for the lottery would you be indifferent?

d: You can choose between having (30x30, 4 hours, 1 hour, 0.25 deg, 30%, EDP/AOA) for sure, or a **##** chance of getting (1x1, 5 minutes, 1 minute, 0.005 deg, 100%, EDP/AOA/Turbulence) and a **1-##** chance of getting (50x50, 12 hours, 2 hours, 0.5 deg, 5%, EDP/AOA). At what probability for the lottery would you be indifferent?

e: You can choose between having (10x10, 6 hours, 20 minutes, 75%, 95%, EDP) for sure, or a **##** chance of getting (1x1, 5 minutes, 1 minute, 100%, 100%, EDP) and a **1-##** chance of getting (50x50, 12 hours, 2 hours, 70%, 5%, EDP). At what probability for the lottery would you be indifferent?

f: You can choose between having (20x20, 40 min, 30 min, 0.5 deg, 60%, EDP/AOA/Turbulence) for sure, or a **##** chance of getting (1x1, 5 minutes, 1 minute, 0.005 deg, 0.005 deg, EDP/AOA/Turbulence) and a **1-##** chance of getting (50x50, 12

hours, 2 hours, 0.005 deg, 5%, EDP/AOA/Turbulence). At what probability for the lottery would you be indifferent?

##: (Probability sequence: 45%, 10%, 35%, 20%, 25%)

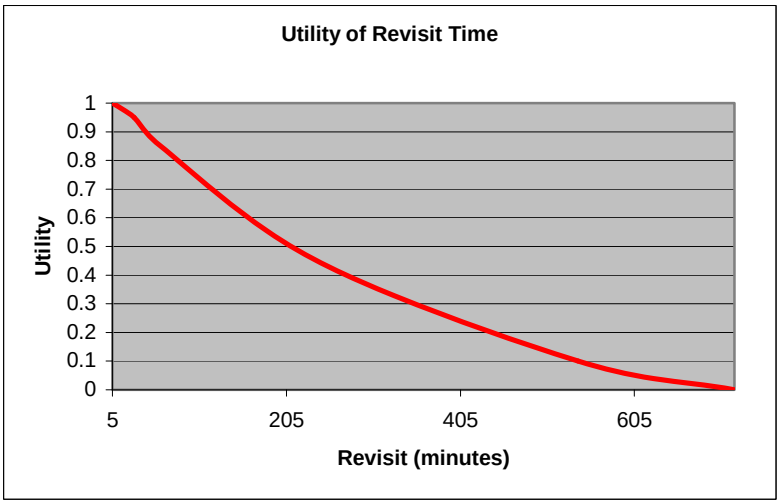
B.2.2 Preferential Independence Questions and Results

						Which Do You Prefer?									
						OR									Selection
Given Conditions						Selection 1					Selection 2				Chosen
Latency	50 min	AOA Accuracy	.25 deg	Inst. Global Coverage	50%	Spatial Resolution	10 X 10	AND	Revisit Time	120 min.	Spatial Resolution	35 X 35	AND	Revisit Time	50 min.	1
AOA Accuracy	.25 deg	Inst. Global Coverage	50%	Spatial Resolution	25 X 25	Revisit Time	120 min.	AND	Latency	20 min.	Revisit Time	15 min.	AND	Latency	40 min.	1
Inst. Global Coverage	50%	Spatial Resolution	25 X 25	Revisit Time	360 min.	Latency	20 min.	AND	AOA Accuracy	0.08 deg	Latency	40 min.	AND	AOA Accuracy	0.01 deg	2
Spatial Resolution	25 X 25	Revisit Time	360 min.	Latency	50 min	AOA Accuracy	0.01 deg	AND	Inst. Global Coverage	20%	AOA Accuracy	0.08 deg	AND	Inst. Global Coverage	40%	1
Revisit Time	360 min.	Latency	50 min	AOA Accuracy	.25 deg	Inst. Global Coverage	40%	AND	Spatial Resolution	35 X 35	Inst. Global Coverage	20%	AND	Spatial Resolution	10 X 10	2
Revisit Time	360 min.	AOA Accuracy	.25 deg	Inst. Global Coverage	50%	Spatial Resolution	35 X 35	AND	Latency	20 min.	Spatial Resolution	10 X 10	AND	Latency	40 min.	1
Revisit Time	360 min.	Latency	50 min	Inst. Global Coverage	50%	Spatial Resolution	35 X 35	AND	AOA Accuracy	0.01 deg	Spatial Resolution	10 X 10	AND	AOA Accuracy	0.08 deg	1
Spatial Resolution	25 X 25	Latency	50 min	Inst. Global Coverage	50%	Revisit Time	120 min.	AND	AOA Accuracy	0.01 deg	Revisit Time	15 min.	AND	AOA Accuracy	0.08 deg	1
Spatial Resolution	25 X 25	Latency	50 min	AOA Accuracy	.25 deg	Revisit Time	120 min.	AND	Inst. Global Coverage	60%	Revisit Time	15 min.	AND	Inst. Global Coverage	20%	2
Spatial Resolution	25 X 25	Revisit Time	360 min.	AOA Accuracy	.25 deg	Latency	30 min.	AND	Inst. Global Coverage	20%	Latency	60 min.	AND	Inst. Global Coverage	60%	1

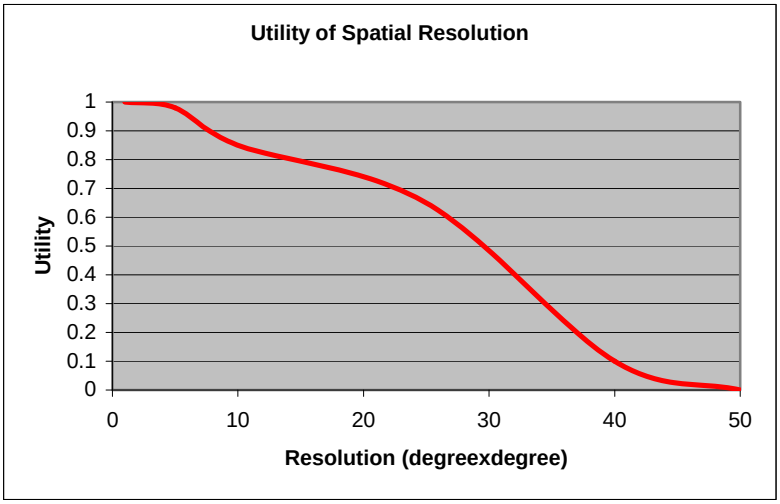
Inst. Global Coverage	50%	Spatial Resolution	25 X 25	Revisit Time	360 min.	Latency	20 min.	AND	EDP Accuracy	80%	Latency	40 min.	AND	EDP Accuracy	80%	1
Revisit Time	360 min.	Latency	50 min	Inst. Global Coverage	50%	Spatial Resolution	35 X 35	AND	EDP Accuracy	90%	Spatial Resolution	10 X 10	AND	EDP Accuracy	80%	1
Spatial Resolution	50 X 50	Latency	120 min.	AOA Accuracy	0.5 deg	Revisit Time	120 min.	AND	Inst. Global Coverage	60%	Revisit Time	15 min.	AND	Inst. Global Coverage	20%	2
AOA Accuracy	0.5 deg	Inst. Global Coverage	5%	Spatial Resolution	50 X 50	Revisit Time	120 min.	AND	Latency	20 min.	Revisit Time	15 min.	AND	Latency	40 min.	1
Revisit Time	720 min.	Latency	120 min.	Inst. Global Coverage	5%	Spatial Resolution	35 X 35	AND	AOA Accuracy	0.01 deg	Spatial Resolution	10 X 10	AND	AOA Accuracy	0.08 deg	1
Spatial Resolution	50 X 50	Revisit Time	720 min.	Latency	120 min.	AOA Accuracy	0.01 deg	AND	Inst. Global Coverage	20%	AOA Accuracy	0.08 deg	AND	Inst. Global Coverage	40%	1
Latency	120 min.	AOA Accuracy	0.5 deg	Inst. Global Coverage	5%	Spatial Resolution	10 X 10	AND	Revisit Time	120 min.	Spatial Resolution	35 X 35	AND	Revisit Time	50 min.	1
Revisit Time	720 min.	Latency	120 min.	AOA Accuracy	0.5 deg	Inst. Global Coverage	40%	AND	Spatial Resolution	35 X 35	Inst. Global Coverage	20%	AND	Spatial Resolution	10 X 10	2
Inst. Global Coverage	5%	Spatial Resolution	50 X 50	Revisit Time	720 min.	Latency	40 min.	AND	AOA Accuracy	0.08 deg	Latency	20 min.	AND	AOA Accuracy	0.01 deg	2
Revisit Time	720 min.	AOA Accuracy	0.5 deg	Inst. Global Coverage	5%	Spatial Resolution	35 X 35	AND	Latency	20 min.	Spatial Resolution	10 X 10	AND	Latency	40 min.	1
Spatial Resolution	50 X 50	Revisit Time	720 min.	AOA Accuracy	0.5 deg	Latency	30 min.	AND	Inst. Global Coverage	20%	Latency	60 min.	AND	Inst. Global Coverage	60%	1
Spatial Resolution	50 X 50	Latency	120 min.	Inst. Global Coverage	5%	Revisit Time	120 min.	AND	AOA Accuracy	0.01 deg	Revisit Time	15 min.	AND	AOA Accuracy	0.08 deg	1
Inst. Global Coverage	5%	Spatial Resolution	50 X 50	Revisit Time	720 min.	Latency	20 min.	AND	EDP Accuracy	80%	Latency	40 min.	AND	EDP Accuracy	90%	1
Revisit Time	720 min.	Latency	120 min.	Inst. Global Coverage	5%	Spatial Resolution	35 X 35	AND	EDP Accuracy	90%	Spatial Resolution	10 X 10	AND	EDP Accuracy	80%	1

B3 Single Attribute Preferences

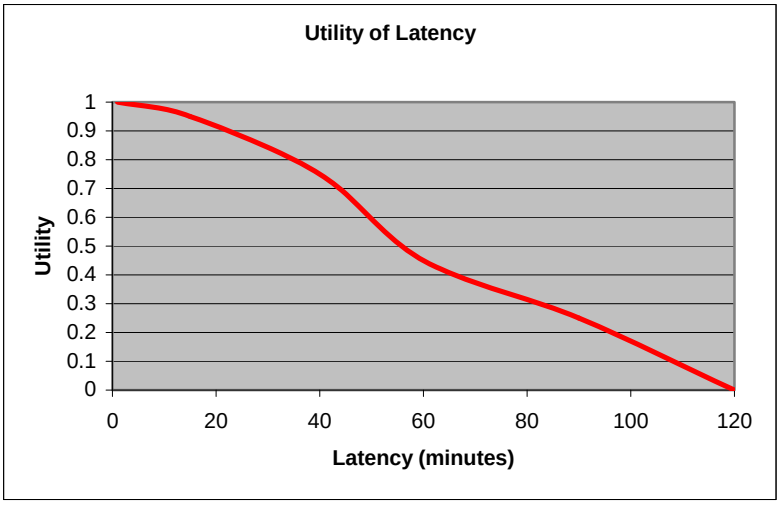
B.3.1 Spatial Resolution



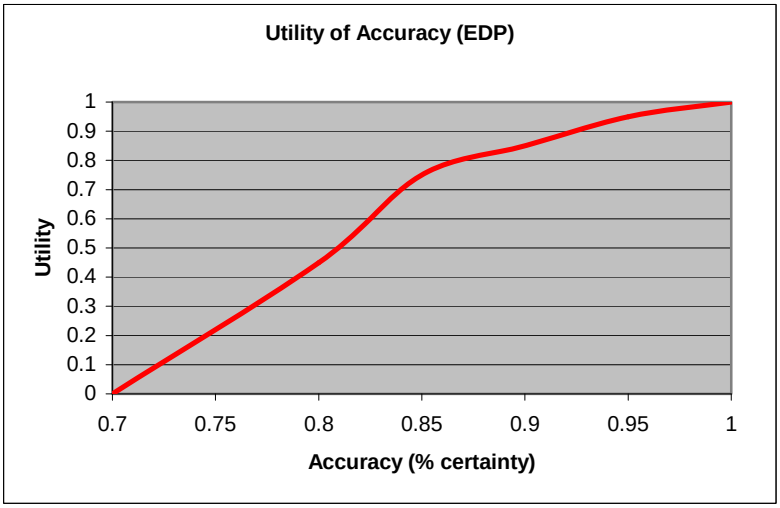
B.3.2 Revisit Time



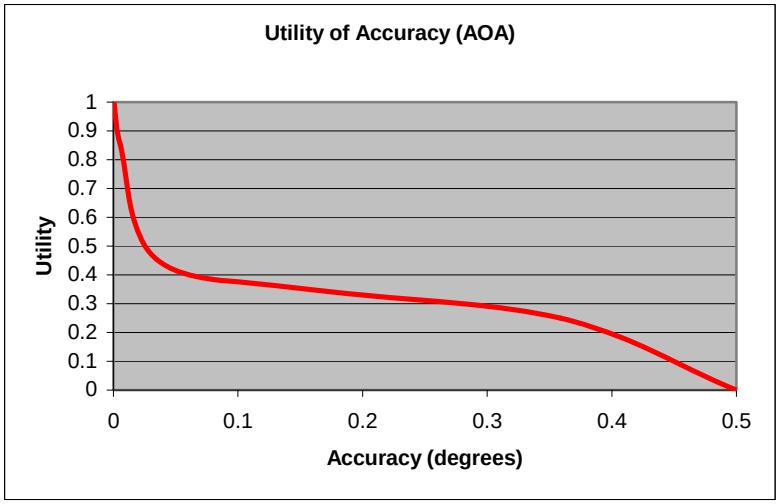
B.3.3 Latency



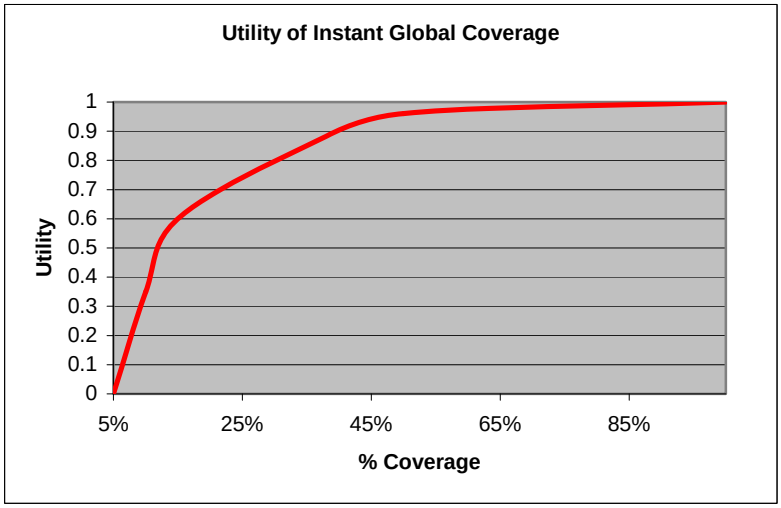
B.3.4 EDP Accuracy



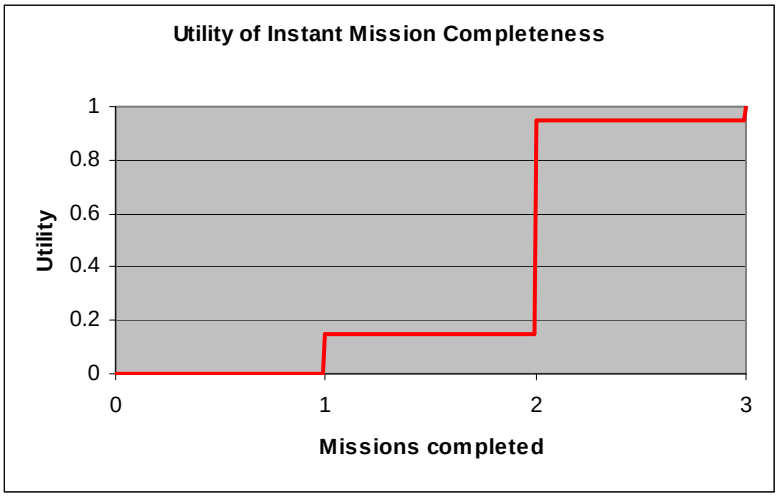
B.3.5 AOA Accuracy



B.3.6 Instantaneous Global Coverage



B.3.7 Mission Completeness



Appendix C
B-TOS Requirements Document

Requirements were derived from the B-TOS architecture analysis. Although the focus of the B-TOS effort was architectural modeling, the resulting analysis enabled a simple, and traceable, set of requirements as listed below. Figure 1 depicts the three-tiered structure of these requirements.

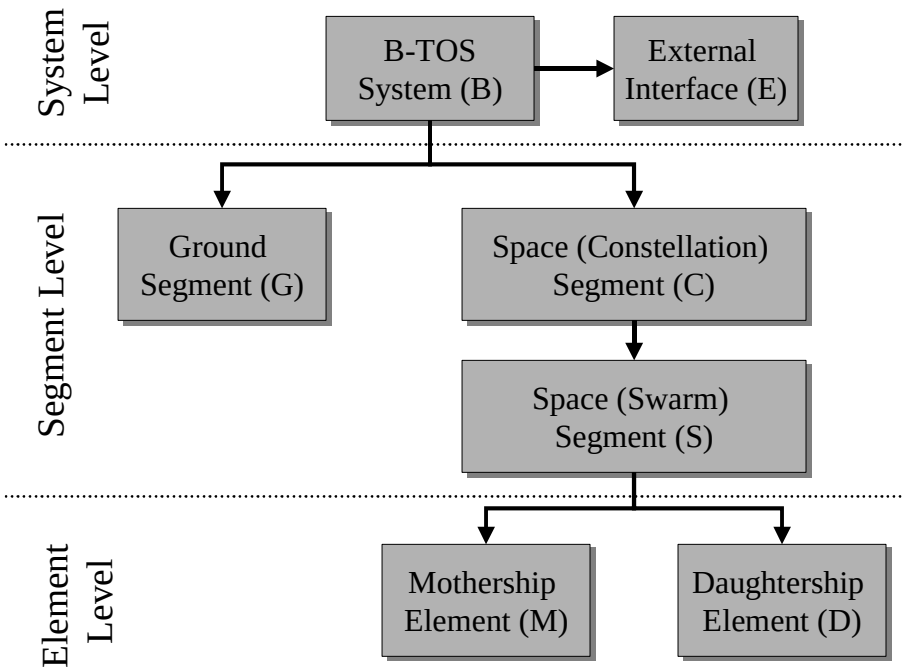


Figure 1: Requirements Structure

B-TOS System Level Requirements

B-1. The B-TOS system shall have the capability to collect data from the topside of the ionosphere below 1100 km to produce an Electron Density Profile (EDP).

B-2. The B-TOS system shall have the capability to determine the Angle of Arrival (AOA) of ground-based beacon transmissions between 30 MHz and 100 MHz.

B-3. The B-TOS system shall have the capability to characterize radio reflections from the topside ionosphere to locate and measure large-scale ionosphere turbulence.

B-4. The B-TOS system shall have the capability to meet Payload B power, thermal, command and data handling requirements.

B-5. The B-TOS system shall be designed to use a launch vehicle manufactured and launched in the United States.

B-6. The B-TOS system shall be designed for an operational lifetime of no less than 5 years.
B-7. The B-TOS system will use TDRSS as its ground communication link.

B-TOS External Interface Requirements

E-1. The B-TOS space system will be capable of communicating with TDRSS
E-2. The B-TOS space system shall be compatible with current U.S. launch vehicles.
E-3. The data from the B-TOS system shall provide properly formatted input for the AFRL/VSB ionospheric forecasting model.
E-4. The B-TOS space system shall receive signals from AFRL-designated ground beacons to perform AOA mission.

B-TOS Segment Level Requirements

Space (Constellation) Segment

C-1. The constellation shall have one plane.
C-2. The constellation shall have one swarm per plane.
C-3. The constellation shall be in an orbit at 1100 km (+/- 50 km).
C-4. The constellation orbits will be inclined at 63.4°.
C-5. The constellation shall be capable of transferring data to an ionospheric forecasting model less than 5 minutes after collection.
C-6. The constellation shall provide 1% (+/- .15%) global coverage within the 130 second duty-cycle to collect one set of ionospheric measurements.
C-7. The constellation shall provide reoccurring coverage of any spot on the globe within 500 minutes.

Space (Swarm) Segment

S-1. Each swarm shall have ten satellites consisting of 1 mothership and 9 daughterships.
S-2. Each swarm shall have an outer Hill's Radius of 8.75 km (+/- 0.10 km).
S-3. Each swarm shall have full intra-swarm communication between each daughtership and the mothership at all times.
S-4. Each swarm shall have at least one communication link to TDRSS.
S-5. Each swarm shall have a measurement set spatial resolution of 7.3 square degrees (+/- 0.2 sq. deg.).
S-6. Each swarm shall have an AOA mission accuracy not worse than 0.0030 degrees (+/- 0.0005 deg.).

Ground Segment

G-1. The operations center shall perform mission scheduling.
G-2. The operations center shall communicate to each swarm through TDRSS.

G-3. The operations center shall receive space segment telemetry data.
G-4. The operations center shall receive space segment payload data.
G-5. The operations center shall process data into AFRL usable format.
G-6. The operations center shall provide swarm command and control.
G-7. The operations center shall provide space segment engineering trending and analysis.

B-TOS Satellite (Element) Level

Mothership Element

M-1. The mothership shall have a communication subsystem capable of sending data at 5 Mbps and receiving data at 100 kbps with the ground via TDRSS' S-band single access antennas at 10^{-6} bit error rate.
M-2. The mothership shall have a communication subsystem capable of receiving continuous data at 1.2 Mbps from each daughtership in the swarm.
M-3. The mothership shall have a communication subsystem capable of sending command data at 10 kbps to each daughtership in the swarm.
M-4. The mothership shall be capable of compressing payload data at least at a ratio of 3:1.
M-5. The mothership shall be capable of performing all payload missions.
M-6. The mothership shall be capable of meeting payload B requirements.

Daughtership Element

D-1. The daughtership shall have a communication subsystem capable of sending data at 1.2 Mbps and receiving data at 10 kbps with the mothership.
D-2. The daughtership shall be capable of receiving measurements for AOA and turbulence missions.

Appendix D
B-TOS Payload Requirements

Swarm Satellite Mission
BTOS Payload Requirements
30 April 2001

	Central Element (Mothership)	Remote Elements (Daughterships)
Attribute	value	value
Peak Power	109W	53W
Orbit Avg Power	64W	14W
Mass	36 kg	16.1 kg
Physical Size	**2x** 26.5x23.0x20.3 cm^3 + **6x** 10m whip antennae + **6x** 13.2x6.8x25.6 cm^3 antenna deployers + 4424 cm^3	26.5x23.0x20.3 cm^3 + **4x** 10m whip antennae + **4x** 13.2x6.8x25.6 cm^3 antenna deployers + 1311 cm^3
Short Range Comm	Payload commanding to remotes tbd (low)	1.2 Mbs to central element
Long Range Comm	1Mbs	none
Timing Knowledge	As required for AOA determination (see wsb notes)	As required for AOA determination (see wsb notes)
Position Knowledge	10 m (x,y,z)	Relative knowledge required for AOA determination (see wsb notes)
Position Control	+/- 50 km	Relative contrl required to maintain AOA accuracy and FOV (see wsb notes)
Pointing Knowledge	.05 degrees	0.5 degrees
Pointing Control	+/- 5 degrees	+/- 5 degrees

Kbs = kilobits per second
Mbs = megabits per second

Note: The 1311cm^3 boxes on the remote elements need unobstructed nadir FOV, the 6 whip antennae on the central element need to be orthogonally arranged and the four whip antennae on the remotes need to be orthogonal and planar. There are no positioning requirements on the central element boxes.

Appendix E
Spacecraft Design

An exercise was done to create a preliminary design of the mothership vehicle, to both check the assumptions made in the architecture development and to take a first step towards defining the real vehicle.

A greatly simplified Integrated Concurrent Engineering (ICE) methodology was used. The vehicle was divided into functional subsystems, and several budgets were defined, some of which (power, volume) corresponded to a system, and some of which (mass, cost) did not. The interactions of the subsystems were captured on an N-squared diagram, and decisions were made as to the depth of analysis desired for each subsystem. The requirements for, staffing of, and analysis technique to be used in each subsystem are given on Table D-1. The payload requirements provided by the customer (Bill Borer) are in Table D-2.

The N-squared diagram is below as Table D-3. It is a "counter-clockwise" design structure matrix (DSM), with information passing from the sub-system in the column to the one in the row. It is a rather dense figure, with various interactions captured by different codes. 'r' indicates a requirements flow; 'b' a budget impact, and 'k' a "kickback", i.e. a feedback that may be inactive unless a problem occurs. Then a budget (e.g. weight or power) might "kick" the subsystems to save weight or power because the overall vehicle has a problem. Other specific dependencies are shown on the chart.

The N-squared diagram was rearranged to reveal some interesting structure in the interactions. Note the linkage between Comm and C.D.H; the propulsion, thermal and comm. interactions with configuration; and the pervasive (but hopefully inactive) "kickbacks" if budgets are overrun.

A mini-ICE exercise was done in class with one-person teams. SMAD techniques were used to size each subsystem and provide the input to related systems and to the budget. The team was small enough that unstructured information flow (i.e. oral and whiteboard) worked reasonably well. A complete iteration was done on the design. Mass, power, and volume budgets were tallied, the totals were found to change several subsystems, and these were changed and budgets re-tallied.

The results are on the Table E-4 below. Cost, weight and power were all found to vary slightly from the original BTOS architecture assumptions. Weight was up 17%, and power down 21%, from estimates made as part of the architecture definition. The cost of $45M for the mothership alone was a significant fraction of the total spacecraft budget (from the architecture study, $101M). No cost or weight-cutting iterations were performed, so the variations could be mitigated; in any case they were not surprising. No "show-stopping" problems were revealed in the preliminary design, although the comm. requirements through TDRS were thought to be somewhat unrealistic (they would bog down the TDRS system, competing with national assets such as ISS and STS). Also, the solar panel area required is approaching the area available on one side of the spacecraft, suggesting a "power crunch" if the vehicle's power needs increased.

Table E-1: Subsystems for design exercise

Sub-system	Requirement	Approach	Who
Power	Full ops at end of life, peak and avg	Size battery and solar cell	Carol
Thermal	Acceptable temp range at eol, temp range	Energy balance	Adam
Payload	List from customer	Set requirements for other systems	
Comm	Comm through TDRS and with all daughters	Link budget	Scott, Brandon
Attitude	Set by payload	Select and size sensors, wheels, and motors	Nathan
Structure	Not fail or resonate	15% mass fraction budget	Hugh
C.D.H	Support operations, survive environment	Recall ops scenarios, develop link budget inputs, select and size computers and recorders	Qi, Dan
Propulsion	Provide deltaV and max impulse to support ops scenarios	Select and size motors, possibly combined with attitude, consider drag, deorbit, margin, NOT differentials)	Brian, Hugh
Configuration	Fit in launch vehicle and config in 3D	Sketch or CAD	Sandra
Mass	Launchable	Sum up systems' masses	Hugh
Reliability	No single-point failures of vulnerable systems	Check batteries, computers, sensors, thrusters, thermal	Dan
Cost	Not exceed reasonable cost	SMAD cost estimating relationships	Michelle

Table E-2: Detailed Payload Requirements

Swarm Satellite Mission
BTOS Payload Requirements
30 April 2001

	Central Element (Mothership)	Remote Elements (Daughterships)
Attribute	value	value
Peak Power	109W	53W
Orbit Avg Power	64W	14W
Mass	36 kg	16.1 kg
Physical Size	**2x** 26.5x23.0x20.3 cm^3 + **6x** 10m whip antennae + **6x** 13.2x6.8x25.6 cm^3 antenna deployers + 4424 cm^3	26.5x23.0x20.3 cm^3 + **4x** 10m whip antennae + **4x** 13.2x6.8x25.6 cm^3 antenna deployers + 1311 cm^3
Short Range Comm	Payload commanding to remotes tbd (low)	1.2 Mbs to central element
Long Range Comm	1Mbs	none
Timing Knowledge	As required for AOA determination (see wsb notes)	As required for AOA determination (see wsb notes)
Position Knowledge	10 m (x,y,z)	Relative knowledge required for AOA determination (see wsb notes)
Position Control	+/- 50 km	Relative contrl required to maintain AOA accuracy and FOV (see wsb notes)
Pointing Knowledge	.05 degrees	0.5 degrees
Pointing Control	+/- 5 degrees	+/- 5 degrees

Kbs = kilobits per second
Mbs = megabits per second

Note: The 1311cm^3 boxes on the remote elements need unobstructed nadir FOV, the 6 whip antennae on the central element need to be orthogonally arranged and the four whip antennae on the remotes need to be orthogonal and planar. There are no positioning requirements on the central element boxes.

Table E-3: N-squared diagram

	Payload	Attitude	C.D.H	Comm	Therm.	Prop.	Config	Power	Mass	Structure	Reliability	Cost
Payload	X											
Attitude	know .05 deg point 5 deg	X		r				k	k		k	k
C.D.H.	1 Mbs		X	1.2 Mbs per daught.				k			k	
Comm	1.2 Mbs each from daughter	facing	BPS and BER for ground link	X			available places	k	k		k	k
Thermal		facing	r	r	X	r	geometry	solar cell props tot pow	k			k
Propulsion		mom. dump/ time				X	available places	available power bogey	k		k	k
Config.	bills memo	facing /shape		antenna place.	surfaces for heat/ cool	desired thruster place	X	b				
Power	109 peak 64 ave	b	b	b	b	b		X	k		k	k
Mass	36kg	b	b	b	b	b		b	X			
Structure									total mass	X		
Reliability		reliability info	reliability info	reliability info		reliability info		reliability info			X	
Cost	info	info	info	info	Info	info		info	total mass	info		X

Table E-4: Resulting system

Sub-system	Spec	Power	Mass	Cost
Payload	6 omni antenna plus transceivers	64W	36kg	N/A
Attitude	3-axis momentum wheels	20W	7kg	$9.8M (±4.4)
C.D.H.	Computers plus data storage	14W	5kg	$6M (±2.4)
Comm	0.5m diameter antenna	10W	20kg	$3M (±0.6)
Thermal	$0.32m^2$ radiator plus radiative paint	1.3W	4.5% dry mass	$8M (±1.4)
Propulsion	12 PPT thrusters	40W	20kg dry plus 7.30kg fuel	$6.5M (±1.5)
Configuration	Cylinder (D=H=1.5m)	N/A	27kg (structure plus thermal)	$1.6 (±1)
Power	$2.5m^2$ Si body mounted solar arrays 4 NiCd batteries	Total Power Req: 150W EOL Daylight Power Produced: 285W	33.5kg	$16.7M (±7.1)
Mass	Sum of all systems	N/A	Totals: 185kg dry 193kg w/ fuel 208kg boosted	N/A
Reliability	N/A	N/A	N/A	N/A
Cost	SMAD cost estimating relationships	N/A	N/A	Totals: S/C $45M (±19)

Appendix F
Interferometric Considerations for Satellite Cluster Based HF/LVHF Angle of Arrival Determination

Bill Borer
05 May 2001

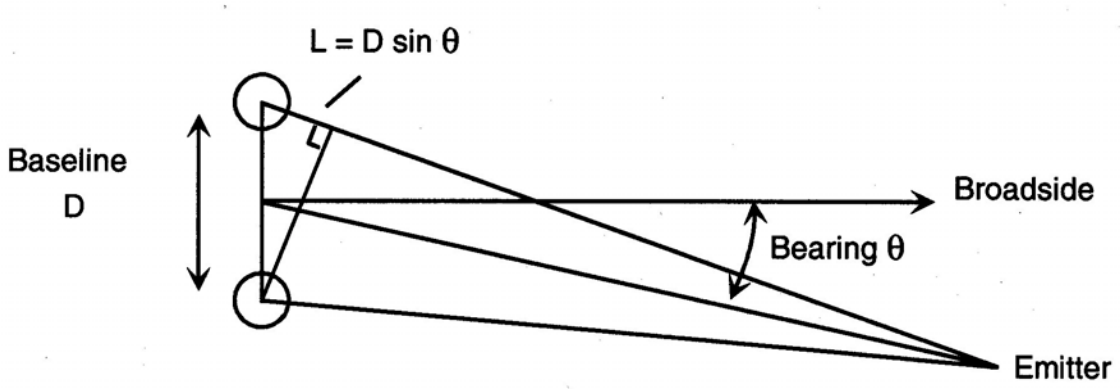


Figure 1: Two Element Interferometer

$$\sin\theta = \frac{L}{D} = \frac{\lambda\phi}{2\pi D} \qquad (1)$$

$\phi \equiv$ difference in phase detected at the two receivers

A) Error in Bearing Determination:

$$\cos\theta\, d\theta = \frac{1}{2\pi}\left\{\frac{\lambda}{D}d\phi - \frac{\lambda\phi}{D^2}dD\right\} \qquad (2)$$

Treat the limiting case where phase error is all due solely to timing measurement error.

$$d\phi = \frac{2\pi dT}{\text{period}} = 2\pi f\, dT = 2\pi\frac{c}{\lambda}dT \qquad (3)$$

$d\phi$ = error in phase difference measurement

$dT \equiv$ error in time difference measurement

Equation (3) is valid for infinitely precise phase measurements. Derived errors are therefore lower limits to those physically attainable.

$$\therefore d\theta = \frac{1}{2\pi \cos\theta}\left\{\frac{2\pi c dT}{D} - \frac{\lambda\phi}{D^2} dD\right\} \qquad (4)$$

$$= \frac{1}{2\pi \cos\theta}\left\{\frac{2\pi c dT}{D} - \frac{2\pi \sin\theta}{D} dD\right\} \qquad (5)$$

$$= \frac{c}{D\cos\theta} dT - \frac{\tan\theta}{D} dD \qquad (6)$$

c/D = (1/propagation time across baseline) (7)

Equation (6) will have added to it a term due to error in the relative position perpendicular to the baseline, orientational error. This term is small and constant in bearing angle.

Bearing error is due to timing and positioning error.
Bearing error is a function of bearing angle.
Bearing error is independent of wavelength.

θ	1/cos(θ)	tan(θ)	%FOV
0	1	0	
15	1.04	0.27	17
30	1.15	0.58	33
45	1.41	1.00	50
60	2.00	1.73	67
75	3.86	3.73	83
80	5.76	5.67	89
84.3	10.1	10	94
85	11.5	11.4	
86	14.3	14.3	
87	19.1	19.1	
88	28.7	28.6	
89	57.3	57.3	

For dT = 1 nanosecond, dD = 0.1 m and a baseline of 100km,

$$\frac{c}{D} dT = .003\ milliradians$$

$$\frac{dD}{D} = .001\ milliradians$$

.1 milliradian accuracy goal appears achievable over 94% of the FOV.

A 100 meter baseline would have three orders of magnitude less accuracy which is still of the order of 1 milliradian accuracy at broadside, .057 degrees.

B) Spacing of Null Lines:

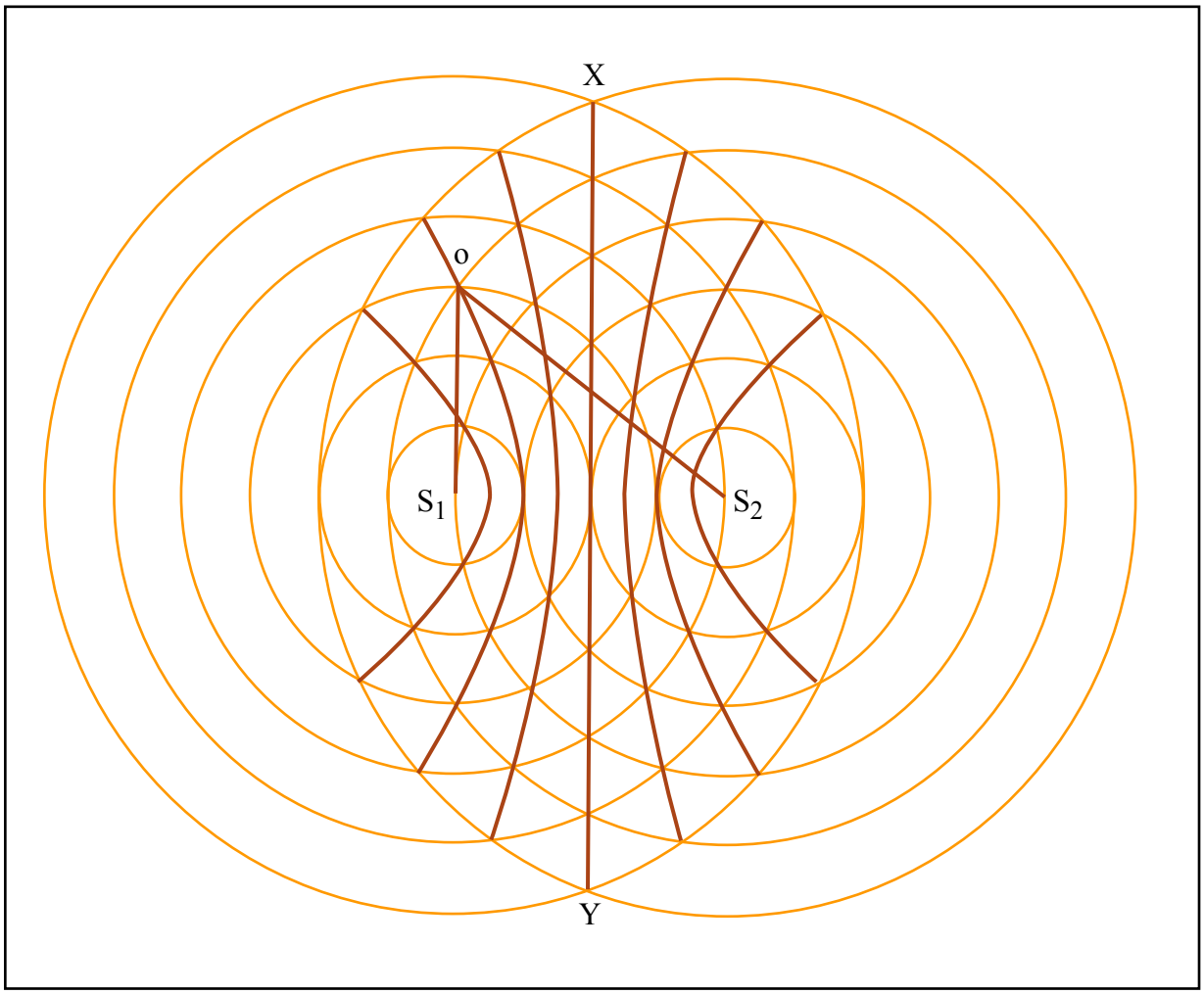


Image by MIT OpenCourseWare.

			D			
			100 km	**10 km**	**1 km**	**100 m**
Period	**Frequency**	**Wavelength**				
333 nsec	**3 MHz**	**100 m**	10^{-3}	10^{-2}	.1	1
33.3 nsec	**30 MHz**	**10 m**	10^{-4}	10^{-3}	10^{-2}	.1
3.33 nsec	**300 MHz**	**1 m**	10^{-5}	10^{-4}	10^{-3}	10^{-2}

Table of R = λ/D

Period (nsec)	**f (MHz)**	**λ (m)**
333	**3**	100
100	**10**	30
33.3	**30**	10
20	**50**	6
17	**60**	5
10	**100**	3
3.3	**300**	1

$$\phi = \frac{2\pi}{R}\sin\theta \qquad (9)$$

$$\frac{\partial\phi}{\partial\theta} = \frac{2\pi}{R}\cos\theta \qquad (10)$$

$$\frac{\partial\theta}{\partial\phi} = \frac{R}{2\pi}\frac{1}{\cos\theta} \qquad (11)$$

$$null\ spacing = 2\pi\frac{\partial\theta}{\partial\phi} = \frac{R}{\cos\theta} \qquad (12)$$

θ	1/cos(θ)	tan(θ)	Null Spacing for R=.1 (degrees)
0	1	0	5.73
15	1.04	0.27	5.93
30	1.15	0.58	6.62
45	1.41	1.00	8.10
60	2.00	1.73	11.5
75	3.86	3.73	22.1
80	5.76	5.67	33.0
85	11.5	11.4	66
86	14.3	14.3	82
87	19.1	19.1	109
88	28.7	28.6	164
89	57.3	57.3	328

C) Multi Stage "Vernier" Technique for Utilizing Long Baselines:

Now consider using one baseline, $D_<$, as a coarse acquisition to resolve bearing to within one null spacing of a larger baseline, $D_>$.

$$\Delta\theta_> = R = \frac{\lambda}{D_>} \equiv \text{null spacing factor of larger baseline} \qquad (14)$$

$$d\theta_< = \frac{c}{D_<} dT - \frac{dD}{D_<} \equiv \text{accuracy of smaller baseline} \qquad (15)$$

$$\text{need } d\theta_< \leq \Delta\theta_> \qquad (16)$$

$$\frac{cdT}{D_<} - \frac{dD}{D_<} \leq \frac{\lambda}{D_>} \qquad (17)$$

$$\frac{cdT - dD}{\lambda} \leq \frac{D_<}{D_>} \qquad (18)$$

$$\frac{dT}{period} - \frac{dD}{\lambda} \leq \frac{D_<}{D_>} \qquad (19)$$

for dT = 1 nanosecond, dD = .1m and λ = 3 m,

$$\frac{dT}{period} = .1 \qquad (20)$$

$$\frac{dD}{\lambda} = .033 \qquad (21)$$

$$\therefore D_> \leq 7.5 D_< \qquad (22)$$

A sufficient sequence of baselines would be 100km, 13km, 1.7km, 237m and 31m.

D) Constraints on Shortest Baseline:

Let $d\theta_0$ be the accuracy of three orthogonal antennae on board one spacecraft and $\Delta\theta_1$ be the broadside null spacing of the shortest baseline.

$$d\theta_0 < \Delta\theta_1 \equiv R_1 = \frac{\lambda_{\min}}{D_1} \qquad (25)$$

or

$$D_1 < \frac{\lambda_{\min}}{d\theta_0} \qquad (26)$$

1 degree = 0.017 radians
4 degrees = 0.070 radians

	D_1		
f_{max}	**30 MHz**	**60 MHz**	**100 MHz**
λ_{min}	**10 m**	**5 m**	**3 m**
$d\theta_0$			
.017 radian	588 m	294 m	176 m
.070 radian	143 m	71 m	43 m

Accuracies of these baselines must be better than the accuracy of the orthogonal antennae in order to graduate to a larger baseline.

$$d\theta_1 = \frac{cdT}{D} \qquad (28)$$

For dT = 1 nanosecond, dD = 0.1 m, and D = 43 m:

$$\frac{cdT}{D} = .006 \qquad (29)$$

The accuracy threshold is met.