Multi-level Decision-based System Architecting

by

Arthur N. Guest

B.Sc., Mechanical Engineering, University of British Columbia, 2005

M.S., Space Studies, International Space University, 2006

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

MASTERS OF SCIENCE

Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2011

Signature of Author..................[illegible]......................

Department of Aeronautics and Astronautics

February 4, 2011

Certified by......................................[illegible].........

Edward F. Crawley

Ford Professor of Engineering

Thesis Supervisor

Accepted by..

Prof. Eytan H. Modiano

Associate Professor of Aeronautics and Astronautics

Chair, Committee on Graduate Students

Multi-level Decision-based System Architecting

by

Arthur N. Guest

Submitted to the Department of Aeronautics and Astronautics in partial fulfillment of the requirements for the degree of Master of Science in Aeronautics and Astronautics.

ABSTRACT

Decision-based system architecting represents a complex system as a set of interconnected decisions that a system architect can make about the trade space. Modeling a system using decision-based frameworks allows the architect to not only enumerate and evaluate feasible architectures, but also, to gain insight into how influential each decision is to the overall system. A single-level decision-based model allows an architect to examine the decisions and architectural combinations at a single level of abstraction of the architecture. Through linking of multiple single-level models, each focusing at a different level of abstraction of the architecture, the system architect can gain insight into the decisions and options for system, element, and component designs while maintaining the validity of the decision support data provided by the models. Single and multi-level decision based system architecting is applied to multiple future human spaceflight projects. The architecture of the Lunar Surface System and the mission design for a human Near Earth Object (NEO) mission are examined through single-level models. The habitat design for potential NEO missions is further examined through the application of multi-level decision-based system architecting techniques.

Thesis Supervisor: Edward F. Crawley

Title: Ford Professor of Engineering

Acknowledgements

First and foremost, I would like to sincerely and whole-heartedly thank my amazing wife Lindsey for supporting me, not only in my efforts at MIT, but in everything in our lives together. Without her I would not have accomplished anything. I would also like to profusely thank my advisor Prof. Edward F. Crawley for not only guiding my research at MIT but for providing the time to teach and work with me throughout the process. Kathi Brazil should also be thanked for providing moral support and taking care of almost every administrative detail possible. Additional thanks and acknowledgements go to all of the students, past and present, at MIT whom I had the pleasure of learning from. In particular, I would like to acknowledge Wilfried Hofstetter for the knowledge and wisdom he provided throughout our time at MIT.

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Motivation

As the construction of the International Space Station (ISS) nears completion and as NASA contemplates the retirement of the Space Shuttle, the human spaceflight community has begun to turn its attention to designing the systems required for sending humans to explore destinations beyond Low Earth Orbit (LEO). While many agree that having humans live and work on the surface of Mars is the ultimate goal for human spaceflight in the foreseeable future, there is also an agreement among many that this should not be the next destination for humans beyond LEO [Aug09]. Instead of focusing on solving all the technical challenges required to send humans to the surface of Mars, several other destinations and missions have been proposed as nearer-term stepping stones. Not only do these missions prepare humans to eventually explore the Martian surface, but also they can provide numerous benefits, such as public engagement and scientific knowledge, on their own.

Numerous studies have either been undertaken or are currently being undertaken, which focus on the design of the missions and systems required for human spaceflight projects to destinations beyond LEO. A list of these studies and a brief description of each is given in Table 1.

Table 1. A list of human spaceflight studies to destinations beyond LEO.

Study Name	Description
NASA FY 1988 Case Studies	Investigated three human expeditions to the surface of Mars. [NAS88]
NASA FY 1989 Case Studies	Investigated an effort to emplace a permanent outpost on the surface of Mars starting in 2007. [NAS89]
NASA First Lunar Outpost (1993)	Aimed at understanding the implications of restoring U.S. lunar exploration capability. [NAS93]
NASA Human Lunar Return (1996)	Investigated how to reduce the cost of previous lunar exploration architectures. [NAS96]
NASA Mars DRM 1.0 (1997)	Developed a reference mission for human exploration of Mars. [Hof97]
NASA Mars DRM 3.0 (1998)	Refined DRM 1.0 to improve identified weaknesses. [Dra98]
NASA DPT/NExT (2002)	Created a new integrated vision and strategy for space exploration including missions to several destinations. [NAS02]
NASA ESAS Report (2005)	Investigated a human return to the lunar surface and the development of a new space transportation system. [NAS05]
NASA Mars DRA 5.0 (2009)	Refined and updated previous Mars DRM architectures. [Dra09]
OSTP Augustine Commission (2009)	Investigated the direction of NASA's integrated human spaceflight program. [Aug09]

These studies, and their respective proposed mission designs, can be divided into two sets based on the type of mission they propose and the destination of that mission. The two sets of missions are:

1. Missions to the surface of a planetary body such as the Moon or Mars. These missions focus on allowing humans to live and work on the Martian or lunar surface for extended periods of time. These missions include both a lunar and Martian transportation system to transit the crew and cargo between the surface of the Earth and the surface of the Moon or Mars as well as a surface system that consists of all the assets required to support the crews on the lunar surface.
2. Missions to in-space destinations in deep space (i.e. beyond LEO). These missions do not descend and land on large planetary objects, but instead explore a destination from in-space vehicles. Possible deep space destinations include Lagrange Points (LPs), NEOs, and the vicinity of Mars.

Regardless of the missions and destinations investigated, the majority of these studies have something in common. Each study focuses on developing a single or small set of interesting scenarios and designs for their respective missions. Each of these scenarios is typically based on the knowledge gained from previous studies and attempts to improve on the previous designs. The disadvantages with this approach are:

1. Each study does not fully explore the architectural trade space using the same assumptions and reasoning.
2. Since the new scenarios are based on previous designs, the new designs tend to be evolutionary in nature and preclude the possibility of a "revolutionary shift" to another part of the trade space.
3. When the same engineers and team members are involved in subsequent evolutionary studies, they tend to champion certain concepts and elements.

Therefore, the mission designers of human spaceflight projects would benefit from a capability to comprehensively investigate the entire architectural trade space of a relevant mission instead of being forced to simply develop and update scenarios from previous studies. The challenge is that these missions are extremely complex and in many cases the systems and operations involved in the mission are novel and not well understood.

Mission designers must enumerate and evaluate thousands, if not more, of different options for the mission and make decisions related to which elements to include and how to operate them. Additionally, most of the elements proposed for these missions are complex systems, in and of themselves, with their own set of decisions on which components to include. Since these missions and elements are novel, mission designers cannot draw on previous experience to determine which decisions are most influential and how all of the relevant decisions are connected. With a clearer understanding of the underlying decisions, a mission designer would be better equipped to comprehensively explore the trade space of feasible mission designs and would be able to choose the best possible mission based on the relevant figures of merit.

1.2 Objective

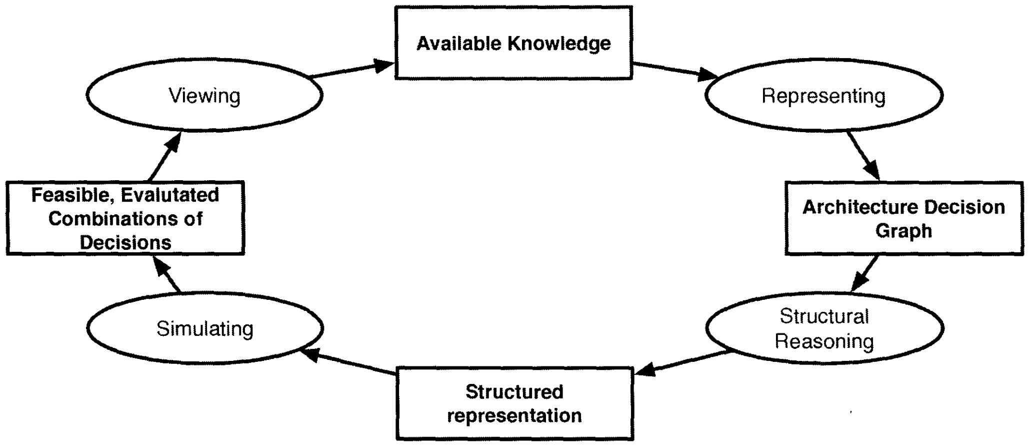
The objective of this research is to provide a framework for formulating and investigating the decisions related to future human spaceflight missions and element designs, thereby allowing a more comprehensive enumeration and evaluation of the options for those specific projects as well as a decision support tool for the system architect to determine which decisions are most influential to the architecture.

1.3 Background

System architecting is the process of transforming the needs and goals of a system into a set of acceptable designs by creating a stable, high-level mapping of functions to forms to embody the concept of the system [Cra07]. In other words, creating a system architecture involves determining the functions involved in ensuring a system meets its desired goals and determining which components or elements will perform the necessary functions. This requires a system architect to understand their system's goals, possible internal functionality, and feasible options for the elements of the system. The architect should enumerate and evaluate all feasible combinations of forms and functions to identify which architectures best meet the desired goals. For systems such as those involved in human spaceflight projects, this process can be complex and challenging since there are a large number of feasible options and the choices of form and function are usually highly interconnected. This creates an extremely large search space that challenges both a human's capability to understand the problem as well as a computer's ability to exhaustively search the space.

Solving difficult search problems, such as those presented by system architecting human spaceflight projects, often depends on being able to effectively represent the system to allow efficient modeling [BL85]. Because the set of needs and goals for a system are transformed into an architecture by making decisions to reduce the candidate space, the process of system architecting is fundamentally a decision making process and it can be effectively represented as a set of interconnected decisions [Sim08]. The assertion that the process of system design is based on decision-making is not a novel concept [Abr65, DAE05, Cat06], however decision-making design methods are often criticized by design theorists [MR02, DAE05] since original decision-based design processes can only be enhanced by a decision-support framework after candidate designs are available. This means that these decision-based methods are only useful for additional refinement of designs that already exist; not for representing a system during the architecture phase of design.

In 2008, Simmons developed the Architecture Decision Graph (ADG) framework as a decision-based system architecture methodology that could be utilized during the early phases of system architecting before detailed designs have been chosen [Sim08]. Simmons' work developed an explicit representation of a system architecture as a set of decisions, and showed that through the use of this representation, an architect could gain useful insight into the architectural candidate space as well as the decisions related to exploring that space. The ADG framework utilizes an iterative process known as the ADG cycle, as shown in Figure 1, which consists of the four steps required in developing a decision support framework for system architecting.



Available Knowledge
Representing
Architecture Decision Graph
Structural Reasoning
Structured representation
Simulating
Feasible, Evalutated Combinations of Decisions
Viewing

Figure 1. The Architecture Decision Graph cycle [Sim08].

The four steps that make up the ADG cycle include:

1. Representing the system: Formulating the architectural problem as a decision problem and formally representing the system as an Architecture Decision Graph (ADG).
2. Structural reasoning about the system: Extracting properties about the decision variables from the structure of the graph itself to provide insight into the system and to aid in simulation of the model.
3. Simulating the model: Transforming the ADG into an executable computer model and running that model in order to enumerate and evaluate all feasible combinations of decisions (i.e. feasible system architectures).

4. Viewing the results: Transforming the results of the simulation into plots that allow a decision-maker or system architect insight into the most interesting architectures and the most influential decisions.

Simmons' ADG framework is the starting point for this research. Through the initial application of the ADG framework to the specific human spaceflight projects that are shown in the case studies, the framework was refined and adapted to develop the next-generation of a decision-based system architecting methodology.

One of the key refinements to the original ADG framework was the introduction of the concept of system architecture levels of abstraction into the process of representing the system as a set of interconnected decisions. The concept of levels of abstraction deals with the fact that any system can be decomposed into varying levels of complexity. For example, a spaceflight mission can be thought of a singular, whole mission (i.e. Level 0) or it can be thought of as a set of spacecraft or similar elements interacting (i.e. Level 1) or it can even be thought of as multiple sets of components of the elements interacting to provide a specific function (i.e. Level 2). Figure 2 shows the multiple layers of abstraction for the a system (Elements 1 and 3 can similarly be decomposed).

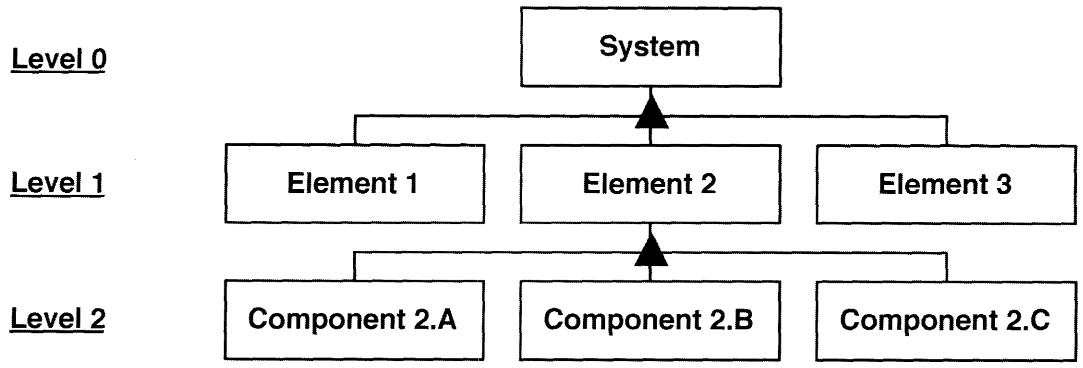


Figure 2. Schematic of the decomposition of a mission system.

This concept and its relation to decision-based system architecting will be further discussed in later chapters, but there is one key fact to remember: the influence of an architectural decision is impacted by how that decision is connected to other decisions in the system. Since the connectivity of a decision is greatly impacted by the level of abstraction it is on, the best way to ensure legitimacy into the insight gained into the influence of a decision through decision-based system architecting is by ensuring all the decisions are on the same level of abstraction.

Ensuring that all the decisions in a decision-based model are on the same level of abstraction allows the development of a single-level decision-based system architecting approach and this is the topic of the first half of this research. However, as noted previously, complex systems, such as those associated with human spaceflight missions, are so large that they often consist of elements which are themselves complex and consist of multiple components. Multi-level optimization and multi-disciplinary optimization are areas of research concerned with how to evaluate and enumerate designs that consist of elements and components that exist on varying levels of abstraction of the system.

Several multi-level multi-disciplinary optimization techniques have been developed to explore large, complex trade spaces that are associated with systems containing multiple levels of elements and components that require designing. Collaborative Optimization (CO) is a two-level MDO technique developed in 1996 [Bra96] that optimizes the design of the components that make up each element (e.g. Level 2 design) by designing to targets related to the variables that connect all of the elements at the together to form a mission (e.g. Level 1 design). Bi-Level Integrated System Synthesis (BLISS) [Sob98], and its derivative, known as BLISS-2000 [Sob03], is a technique that models the system on two levels of abstraction and attempts to enumerate and evaluate the optimal system level solution through the use of weighting factors on the outputs from the element level design.

These techniques have proven applicable to the detail system design of multiple case studies including the detailed design of a reusable launch vehicle [Bro2006], however similar approaches have yet to be applied to decision-based system architecting. The second half of this research focuses on applying the concepts of multi-level modeling to decision-based system architecting.

1.4 Specific Objectives and Outline

In attempting to gain insight into the architectural trade spaces, and their related architectural decisions, of specific human spaceflight projects, this research is focused on the following specific objectives:

- Formalize a single-level decision-based system architecting framework based on an evolution of the original ADG framework. This research is discussed in Chapter Two.
- Apply the single-level framework to an investigation of NASA's proposed Lunar Surface System (LSS) in order to gain insight into potentially interesting LSS architectures and into the influence of relevant element-level decisions (i.e. Level 1 design). This case study is discussed in Chapter Three.
- Apply the single-level framework to an investigation of potential deep space design reference missions in order to gain insight into potentially interesting architectures and into the influence of relevant element-level decisions (i.e. Level 1 design). This case study is discussed in Chapter Four.
- Formalize a multi-level decision-based system architecting framework as an evolution of the single-level modeling framework. This research is discussed in Chapter Five.
- Apply the multi-level framework to a further investigation of deep space design reference missions to gain insight at a component level of abstraction (i.e. Level 2 design). This case study is discussed in Chapter Six.

The final chapter of this thesis, Chapter 7, concludes the research by summarizing the work, discussing relevant recommendations, and by outlining opportunities for future work.

2 Single-Level Decision-Based System Architecting

2.1 Introduction

As part of the Architecture Decision Graph (ADG) framework, Simmons developed a decision-based representation for system architecting that allows insight into the overall system trade space, as well as into the actual decisions that a system architect must make to define the system. This framework attempted to provide for all layers of decision-support (representing, structural reasoning, simulating, and viewing) during the early phases of design associated with system architecting. The initial stage of the research presented in this thesis was to apply the ADG framework to ongoing human spaceflight programs to gain insight into their architectures and decisions. During these case studies, the ADG framework was further developed and evolved to produce the next generation of decision-based system architecture frameworks.

One of the initial and most important evolutions from the ADG framework is related to how the decisions are formulated for a decision-based model. The advantage of decision-based system architecting, compared to optimization techniques that could be used to explore a system's trade space, is that decision-based methods allow not only insight into the trade space, but also insight into how influential each decision is in the overall architecture. However, for this insight into the decisions to be useful, the decisions used in the model must be correctly formulated. This not only refers to how the decisions are phrased, but to ensuring that all the decisions are related to the same level of abstraction within the architecture of the system.

The objective of this chapter is to discuss the theory behind a single-level (of abstraction) decision-based system architecture framework and highlight the evolution from the ADG framework, not just in representing, but also in the other three steps of decision-support.

In order to perform decision-based system architecting, one must go through iterations of the four steps of the framework as shown in Table 2. Each step of the process has one or more tasks that must be undertaken to complete that step. Each cycle through the framework produces new available knowledge allowing the system architect to refine their model.

Table 2. The steps of a decision-based system architecture modeling framework.

Step	Tasks
1. Representing	a. Choosing the property variables (Sec. 2.2.1) b. Formulating the decision variables (Sec 2.2.2) c. Formulating the property functions (Sec 2.2.3) d. Enumerating the logical constraints (Sec 2.2.4)
2. Structural Reasoning	a. Analyzing the system structure (Sec 2.3)
3. Simulating	a. Choosing the simulation strategy (Sec 2.4) b. Implementing the simulation strategy (Sec 2.4)
4. Viewing	a. Choosing the preferred architectures (Sec 2.5.1) b. Investigating the influence of the decisions (sec 2.5.2)

The remainder of the chapter addresses the tasks related to each step of using a decision-based system architecture model with a single section dedicated to each step of the process.

In order to aid in the discussion of the theory behind single-level decision-based system architecting, a simplified human spaceflight example is discussed throughout the chapter. In this example, a spacecraft is to be developed with the goal of supporting human crews in space while they perform scientific experiments in-space. While a real spacecraft has numerous elements that must be considered, for the sake of this example, the spacecraft is to be considered as consisting of two specific elements:

- The habitation module: This element consists of the structure required to provide the necessary pressurized volume to support the astronauts as well as the necessary life-support system to support the desired number of crew for the length of their mission.
- The power system: This element provides the necessary power to the life-support system to ensure the crew's survival for the mission.

For the example, the system architect has been told to investigate options related to the crew size as well as the overall duration of the mission and selecting the preferred habitation module and power system Throughout the remainder of this chapter, a decision-based system architecture model will be developed for this spacecraft beginning with a representation of the system as a decision-based model.

2.2 Representing

The first step in developing a decision-based system architecture model is representing the system. Following from the concept of an Architecture Decision Graph, a system is represented using two types of variables and two types of relations between the variables in order to allow decision-based modeling. These four data types are shown in Table 3.

Table 3. Types of data used in creating a decision-based system architecture model.

Data class	Data type	Description of data types
Variables	Property Variables	Variables of the system used to evaluate an architecture (i.e. metrics or Figures of Merit)
	Decision Variables	Variables of the system controlled by a decision-maker
Relations	Property Functions	Formulas used to calculate the property variables based on the selections made for the relevant decision variables
	Logical Constraints	Propositional statements that specify feasible assignments to two or more decision variables

In order to fully represent the system, four tasks must be accomplished, as shown in Table 2, which align with developing each data type required for the model:

- Choosing the property variables.
- Formulating the decision variables.
- Formulating the property functions.
- Enumerating the logical constraints.

The following sub-sections discuss each task individually and demonstrate the theory by investigating the representation of the simplified human spacecraft.

2.2.1 Choosing the property variables

Since the process of system architecting transforms the needs and goals of the system into a set of feasible architectures, the first task involved in the modeling process is to understand the needs and goals of the system. This allows an informed selection of the property variables. It is the property variables, and by extension the goals of the system, that are used to evaluate each of the feasible architectures to determine which ones are interesting enough for further investigation. In order to perform a complete evaluation, it is necessary to have a "good" set of property variables, or metrics, to capture the value of the system including both the overall benefit and cost of the system.

Previous research related to the ADG framework discussed the importance of selecting the property variables as the first step of representing a system as it allows a system architect to focus their attention on how the system delivers value. However, the paradoxical challenge of choosing a "good" set of property variables, which is not addressed by the original ADG framework, is that:

- The property variables should be high-level enough to capture the true value (i.e. goals) of the system, but these property variables are often abstract and difficult to create quantifiable property variables for (e.g. how does one measure public engagement based on an architecture for a spacecraft).
- The property variables should be low-level enough to ensure a quantitative comparison of the different feasible architectures is possible. However, these property variables make it difficult to know the true difference in the high-level value of the architectures (e.g. how does a difference in the number of EVAs per week that a crew member undertakes affect the amount of scientific knowledge a mission creates).

This challenge of appropriately selecting the evaluation criteria for a model is not unique to decision-based system architecting, but is applicable to any system architecting or early design process. The solution developed for this research is to determine the set of quantifiable "lower-level" property variables by decomposing the stakeholder needs into a coherent set of goals and objectives as shown in Figure 3. This allows the property variables used for decision-based system architecting to be performance markers for concrete objectives; which can flow back to the higher-level stakeholder needs. Through a well-developed stakeholder value network and decomposition of needs to objectives, feasible architectures could then be evaluated based on their "value" to the main entity. This process of decomposing these high-level benefit themes into more quantitative metrics is based on work done by Rebentisch et al [Reb05].

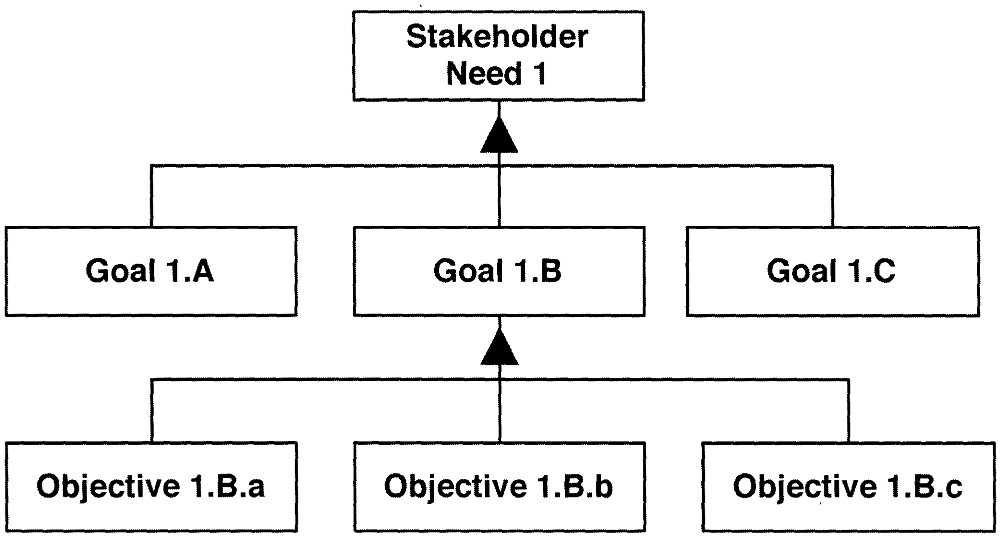


Figure 3. Decomposing stakeholder needs into a coherent set of objectives.

The guidelines for developing a quantitative set of property variables that can be linked to stakeholder needs are as follows:

1. Enumerate the stakeholders of the systems and their needs. The stakeholder needs should encompass both benefit and cost of the system.
2. For each stakeholder need, add further definition by breaking down the need into a set of related goals.
3. For each goal, enumerate the specific objectives that can be satisfied by the system that lead to achieving the goal.
4. For each objective, determine what properties of the system must be measured to determine satisfaction of that object. These are the property variables for the system.

For the example problem of a simplified human in-space vehicle, the property variables must capture both the cost and benefit goals of the system. While there are many possibilities for stakeholder needs related to an in-space mission, the example problem focuses on two major stakeholder needs:

- Providing gains in scientific knowledge
- Ensuring the affordability of the mission.

In decomposing the first need of providing scientific knowledge, it can be seen that this need in itself would be extremely hard to quantify in a realistic manner. Therefore, it should be decomposed into a set of related goals. In this example, one of the goals might be to gain scientific knowledge about the Sun during the mission. There are several objectives that could be met to be able to say that a mission provided adequate knowledge about the Sun. One of these objectives could be performing a set of experiments related to measuring the solar wind. For this objective, the astronauts must have time available to perform the experiments and therefore an appropriate property variable is the number of crew-days available during the mission. The more crew-days available, the more experiments could take place and therefore more scientific knowledge could be gained.

In decomposing the second need of reducing costs, the example problem focuses on a single goal (of which there are several) related to that need: reducing the cost of launching the mission. In thinking about what objectives must be met to reduce the cost, one example is that the overall mass of the in-space vehicle itself should be minimized in order to limit the amount of mass that must be launched. Therefore, the property variable used in the example problem is the mass of the spacecraft that should be minimized to reduce the overall cost of the mission.

Once the two property variables for the system have been chosen (mass and number of crew-days), the system architect can begin thinking about what decisions could be made about the system that might affect these property variables. This leads to the second task of representing the system: formulating the decision variables.

2.2.2 Formulating the decision variables

Once the goals of the systems have been understood and quantified, the next task is to determine the decisions that will have to be made by the system architect to explore the trade space. A decision variable has several feasible alternatives and the system architect has control to select which alternative is chosen. As part of the ADG framework [Sim08], four guidelines were presented to assist in determining the set of decision variables and their alternatives:

1. Set the boundaries of the architecture space under consideration. By limiting the set of decision variables and the number of alternatives for each decision, the model can be kept to a reasonable size.
2. Utilize the property variables to choose decision variables that focus on how the system delivers value. By selecting and understanding the property variables for the system, the architect can ensure that only decisions related to these properties are included. If a decision does not impact any of the property variables, it will not impact the evaluation of the feasible architectures.
3. Capture the architecturally distinguishing decisions. The system architect should focus on the decisions that have the greatest impact on the system and changes the high-level concept of the decision.
4. Keep the problem formulation as simple as possible, but no simpler.

One of the most challenging aspects of developing decision-based system architecture models is in determining which decisions should be included and how they should be formulated. Not only is this important in ensuring that the appropriate trade-offs are investigated, but also in ensuring that the information related to the influence of the decisions provided by the model is valid. Ensuring this validity makes determining the set of decision variables list more challenging then for system architecting models that do not investigate the influence of the decisions. While the original ADG framework provided general guidance on how to select the decision variables, this research has shown that in order to ensure the validity of the decision support information created by the model, further guidance and stricter formulation is beneficial.

One of the most important aspects of selecting the decisions is related to ensuring that all the decisions are on the same level of abstraction for the system. This reasoning is implied by the third guideline presented in the ADG framework related to capturing the architecturally distinguishing decisions (i.e. the model should focus on top-level system decisions), but it has not been discussed explicitly in previous literature [Sim08].

In system architecting, any system and the related decisions can be broken down into levels of abstraction of the system such as system-level decisions (i.e. Level 0), element-level decisions (i.e. Level 1), and component-level decisions (i.e. Level 2) [Cra07]. As developed in the ADG framework and discussed later in this chapter in Section 2.5, one of the aspects related to how influential a decision is in a system is affected by how connected a decision is throughout the model. A decision set made up of multiple levels of decisions will directly and incorrectly impact how influential a decision is. This is because of the tendency of subsystem decisions to be more closely connected with their own subsystem level decisions as compared to other system-level decisions.

Figure 4 shows a schematic of the difference between a single-level decision representation compared to a mixed-level decision representation for an example problem. In the single-level model, all the decisions are formulated at the system level, however, in the mixed representation, one system level decision has been decomposed into component level decisions. The reasoning behind why a certain element of a system may be decomposed is usually that either there is more knowledge or more interest in that specific system level decision and its related subsystems. In this example, as is the case in most systems, the component level decisions related to decomposing system level decision C are strongly connected with each other and not as strongly connected to the other system level decisions.

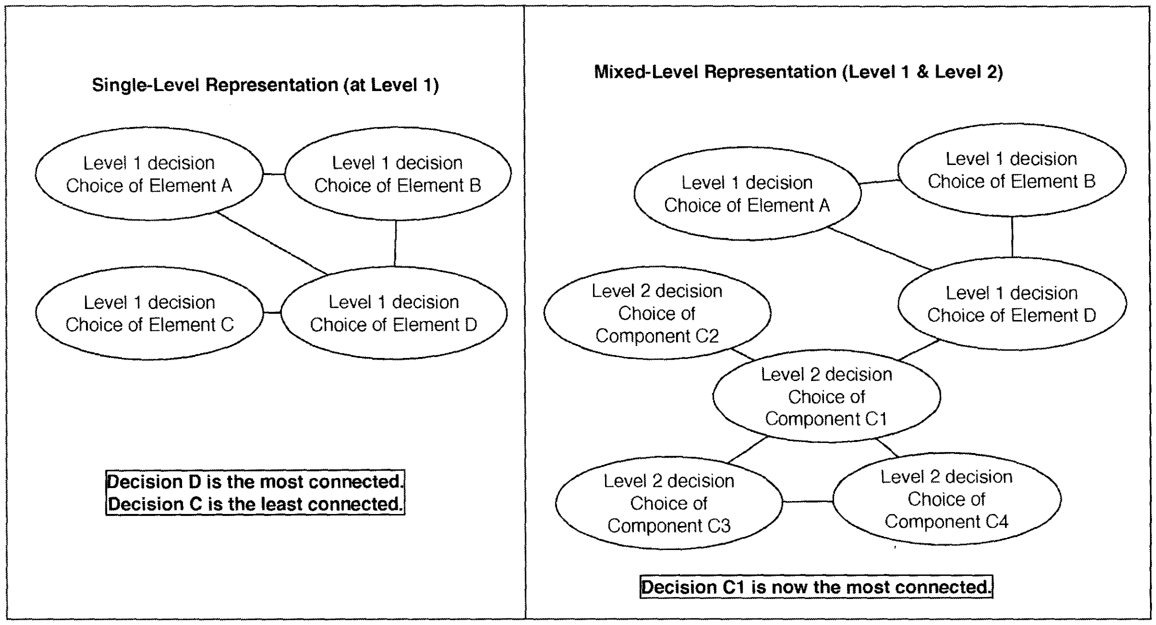

Single-Level Representation (at Level 1)
Level 1 decision Choice of Element A
Level 1 decision Choice of Element B
Level 1 decision Choice of Element C
Level 1 decision Choice of Element D
Decision D is the most connected.
Decision C is the least connected.
Mixed-Level Representation (Level 1 & Level 2)
Level 1 decision Choice of Element A
Level 1 decision Choice of Element B
Level 2 decision Choice of Component C2
Level 1 decision Choice of Element D
Level 2 decision Choice of Component C1
Level 2 decision Choice of Component C3
Level 2 decision Choice of Component C4
Decision C1 is now the most connected.

Figure 4. Single-level decision representation versus mixed-level decision representation.

Through the application of mixed-level decision modeling, the connectivity between the decisions changes. In the single-level model, system-level Decision D is the most connected while system-level Decision C is the least connected; however, in the mixed representation, component level decision C1 is now the most connected. Decomposing system level decision C has artificially produced the perceived influence of the decisions.

In order to ensure that the connectivity and influence of the decisions is valid for the decision support viewing tools, it is necessary to develop the decision list at a single level, such as Level 1 as shown in Figure 2. The decisions at Level 1 relate to the design and operation of each of the major elements of the system. The system could also be modeled at Level 2 with each of the decisions relating to the design and operation of each of the components (or subsystems) that comprise each individual element.

In addition to ensuring that the decisions are on a single-level of abstraction within the system, it is possible to provide further guidelines in how to select and phrase the decision through further application of system architecting principles [Cra07] in order to provide further validity to the decision-support viewing tools. In analyzing the data produced by a decision-based model, it is often important to be able to justify why certain decisions were chosen and why other decisions were not included in the model. While it is correct to say that the decisions not included are not traded by the given model, this does not give sufficient support to defend why certain decisions are included and ensuring that their phrasing is consistent to help support the validity and understanding of the model. In order to develop an architecture for a system, one must specify three aspects of the system:

1. The attributes of the system: these are details that define the capabilities of a system (e.g. the number of crew on a mission).
2. The internal functions that the system must perform: complex systems typically require that two or more internal functions be performed for the system to meet its stated goals.
3. The internal form of the system: these are the specific elements that perform each of the internal functions of the system.

Since system architecting is the process of creating a mapping of forms (i.e elements) of a system to the functions the system has to perform in order to produce value, a set of decisions that a system architect must make is which elements will perform which functions. Therefore a consistent set of form/function decisions would be as follows:

- Decision: What form provides function X?
- Alternatives: None, Element A, Element B, etc.

This set of decisions allows the system architect to develop the mapping required for developing a system architecture and allows the architect to model both the internal functionality and the form-to-function mapping for the system. For every function that the system must perform, there will be an associated decision variable that chooses the element that will perform that function. In some architecture for a system, the internal functionality may not be the same as in other architectures (e.g. a refrigerator may preserve food by chilling it or eradiating it). If this is the case in a system, then for each related decision, one of the alternatives would be "none" and the architecture that employs that alternative would not be capable of performing the related internal function.

If a function is not listed in the decision variables, it means that either no architecture for the system would perform that function or that it is assumed to always be performed and that a single fixed choice for the element is used.

There are also decisions that a system architect must make related to exactly what the concept is. Therefore, the second set of decisions that should be included in a decision-based model is related to choosing the attributes of the system. These decisions can be thought of as decisions related to how the system would be operated. In developing a decision-based model, it is often advisable to investigate the trade space for setting the attributes and the related decisions before defining the form-function decisions. Knowing what attribute decisions will be investigated can provide guidance in choosing the alternative choice for each element.

Based on the two major contributions (ensuring similar level of abstraction and choosing the decisions based on the architecture attributes and form-function mapping), the following guidelines can be added to the original ADG guidelines listed at the beginning of this section:

1. Ensure that all decision variables chosen are on the same level of abstraction of the architecture. By ensuring that all decisions are on the same level, the information concerning the connectivity of the decisions maintains its validity.
2. Focus the decision variables on determining the attributes of the system of interest and on selecting the form-to-function mapping scheme for the system. These decision classes ensure that the final set of decision variables focuses on the architecture the system of interest.

For the example problem of a simplified spacecraft, it has been decided that the decision-based model should be developed at Level 1 of the system (i.e. decisions related to which elements make up the overall system or mission). The task of choosing the decision variables is now defined as choosing a set of both form-function mapping and attributes setting decisions. Based on the initial definition of the system and the choice of property variables, there are assumed to be two attributes of the system that the system architect should set decisions for: the number of crew and the length of the mission.

These decisions and several example alternatives are shown in Table 4. The choice of alternatives for attribute decisions is set by the boundaries of the model.

Table 4. Attribute decisions for the example problem.

Decision Variable	Alternative A	Alternative B
How many crew-members are there?	1	2
What is the duration of the mission?	7 days	14 days

In addition to attribute decisions, there must also be decisions included related to the mapping of form to function for the system. In order to complete the mission, it is assumed that the spacecraft must perform two internal functions that would require elements:

- Provide a habitable volume for the astronauts to work in (simplified in this problem to require an element consisting of a habitable structure and a life support system)
- Provide power to support the mission (simplified in this problem to only consider the power required for the life support system)

Therefore, two decisions should be developed to investigate the mapping of form to function. The alternatives for the habitation element are hypothetical, simplified designs that are optimized for each of the potential concepts of the system (based on the choices of the system parameter decisions). The structure of the habitat is sized assuming it must support a given number of crewmembers and the life support system is sized assuming it must support a given number of crew-days (number of crew multiplied by mission duration). The alternatives for the habitat are enumerated based on all combinations of system parameters and named as seen in the following table.

Table 5. Alternatives for providing the habitation functionality.

Alternative Name	Structure Size	Life Support Capability
Habitat 1.7	Max: 1 crew	Max: 7-crew days
Habitat 1.14	Max: 1 crew	Max: 14-crew days
Habitat 2.14	Max: 2 crew	Max: 14-crew days
Habitat 2.28	Max: 2 crew	Max: 28-crew days

Regarding the power element, for simplicity, the alternatives only include battery options sized to support either a maximum of 7 or 14 days. Based on these assumptions, the list of form-function mapping decisions is shown in Table 6.

Table 6. Form-function mapping decisions for the example problem.

Decision Variable	Alt A	Alt B	Alt C	Alt D
How is in-space habitation provided by the system?	Habitat 1.7	Habitat 1.14	Habitat 2.14	Habitat 2.28
How is power provided by the system?	7-day battery	14-day battery		

Once the decisions set has been developed for the model, the next task is developing and formulating the exact property functions that will connect the four decision variables in the problem to the two property variables that will be used to evaluate the system.

2.2.3 Formulating the property functions

The relation known as a property function uses the characteristics of the chosen alternatives for the relevant decision variables to calculate the property variables. An understanding of the property functions would have developed in the system architect through the selection of the property variables and decision variables, but the next task is to specifically set the formulas for the property functions. Research completed in the development of the original ADG framework showed that there were three types of feasible property functions in terms of how the functions could be written: additively separable, multiplicatively separable, and non-separable. This classification of property functions is useful during the simulation step since the choices of which simulation tools are available for use can depend on what type of property functions exist and what type of property functions a given tool can handle.

One of the challenges in developing the property functions based on the original ADG framework was in formalizing the process to enable transparency into the model and how the architectures are evaluated. Without guidelines on how to specifically and consistently phrase the decision variables, developing the property functions proved challenging. Some decision variables would have to be rephrased to allow a quantitative property function to be developed. However, with a formalized phrasing of the decision variables, it is possible to introduce a specific concept to guide the formulation of the property variables that aids in their development and in the communication of the functions to stakeholders.

In the original ADG framework, each decision variable had several options referred to as alternatives, one of which was selected to form a specific architecture. The concept that has evolved through this research, illustrated in Figure 5, is as follows:

- ***An architecture*** is defined as a combination of ***decision variables***.
- The ***decision variables*** are connected through ***logical constraints***.
- Each ***decision variable*** has a set of feasible ***decision alternatives***.
- Each ***decision alternative*** has a set of specific ***characteristics***.
- The ***characteristics*** of the ***decision alternative*** are the values that are used as inputs into the ***property functions***.
- The ***property functions*** are used to calculate the ***property variables***.
- The ***property variables*** are used to evaluate each ***architecture***.

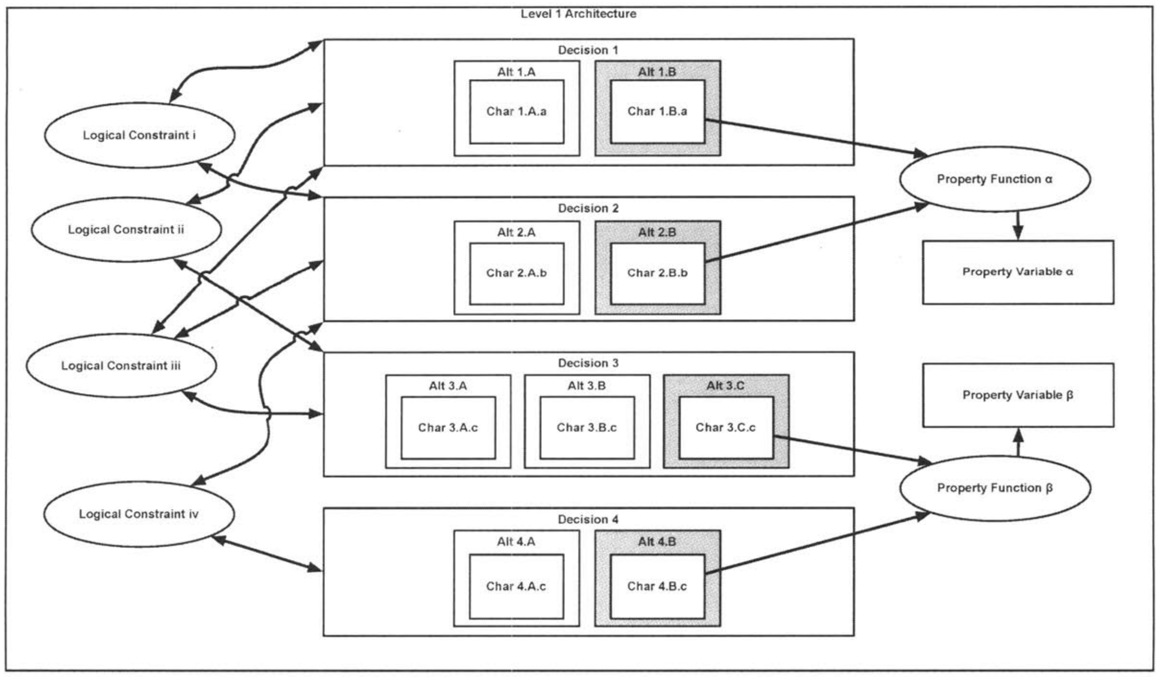


Figure 5. Components of a decision-based model.

(Shading indicates the chosen alterative for each decision in this specific architecture)

The guidelines for developing the decision variables as decisions related to either setting system attributes or form-function mapping provide some clarity for what the required characteristics are. For decision variables related to setting the attributes of the system, the characteristics of the alternative are typically seen in the name of the alternative. For example, the alternative of 1 crew for number of crew-members has the characteristic of having a value of 1 for the number of crew that is used in calculating the property variables. For decision variables related to mapping forms to functions, the characteristics of an alternative are the characteristics of the element that are of important to the property variable (e.g. mass, cost, etc). Not only does this concept assist in formulating the property functions, but it also enables the concept of multi-level modeling as discussed in Chapter 5.

For the example problem of the human spaceflight vehicle, property functions must be developed for both the mass of the system and the number of crew-days enabled by the system. The property function for calculating the overall mass of the system is simply the sum of the mass of the habitat (a characteristic of the chosen alternative) and the sum of the mass of the power system (a characteristic of the chosen alternative). The property function for calculating the number of crew-days enabled is simply the multiplication of the number of crew and the duration of the mission. Therefore, the property functions can be succinctly presented to stakeholders using the following tables.

Table 7. Property functions for the example problem.

Property Variable	Property Function
Mass of System	sum(element masses)
Number of crew-days	Number of crew x Mission duration

Table 8. Characteristics for the example problem decision alternatives.

Decision Variables	Alternatives	Characteristics		
		Number of Crew [-]	Mission Duration [days]	Element Mass [mt]
Number of crew	1	1	-	-
	2	2	-	-
Mission duration	7	-	7	-
	14	-	14	-
Habitation functionality	Habitat 1.7	-	-	1.3
	Habitat 1.14	-	-	1.6
	Habitat 2.14	-	-	2.6
	Habitat 2.28	-	-	2.9
Power provision functionality	7-day battery	-	-	2.0
	14-day battery	-	-	4.0

Once the above three tasks have been completed, the decision-based model can now enumerate and evaluate all combinations of decision alternatives (i.e. architectures) for the system. The problem is that some of these architectures are feasible combinations of alternatives and some of them are infeasible combinations. The next task is to ensure that the only architectures enumerated are feasible architectures through the application of logical constraints.

2.2.4 Enumerating the logical constraints

The logical constraints are the relations in the model that ensure that only feasible architectures are enumerated. They are propositional statements that specify the feasible combinations of alternatives for two or more decision variables. All of the logical constraints must be satisfied in order to ensure an architecture is feasible. In addition to ensuring feasibility, the number of logical constraints impacts the data provided by the model in regards to the influence of a decision variable. The more logical constraints a decision is connected to, the more influential it is since making that given decision impacts the choices for several other decisions in the system.

Given the importance of choosing the logical constraints for the model, guidelines have been created based on the experience of applying decision-based modeling to the case studies presented later in the research. When developing the set of logical constraints for the model, a system architect must think about the three reasons why a logical constraint may exist:

- Logical incompatibilities in the architecture. Certain combinations of decision alternatives are logically incompatible. These are objective constraints.
- Reasoning about the sensibility of the architecture. Certain combinations are feasible, but do not make sense. These are subjective constraints.
- External requirements on the architecture. Certain combinations may be necessary to satisfy the requirements (such as a study's ground rules and assumptions or standards or standard interfaces) set by others. These are subjective constraints.

Logical constraints between decision variables can also exist for a combination of these reasons. For example, certain alternative combinations between two decisions may not be feasible because of logical incompatibly reasoning and certain combinations may not be feasible because of sensibility reasoning. The subjectivity of logical constraints related to sensibility reasoning means that a system architect must discuss these constraints with stakeholders and must ensure that there is agreement for their inclusion in the model. It is recommended that the system architect also acknowledge and discuss logical constraints set by external requirements with stakeholders. It may be beneficial to produce model results for simulations with and without the application of these constraints to show the impact of the given requirements on both the trade space of architectures as well as the perceived influence of each decision.

For the example problem of a simplified human spacecraft, there are four logical constraints to be modeled in the system. The logical constraints are shown below as if-then statement that must be satisfied to ensure a feasible combination of decisions. In order to assist in reading these statements, the list of decision variables and alternatives is reproduced in Table 9.

Table 9. Decision variables for the example problem.

Decision Variables	Alt A	Alt B	Alt C	Alt D
#crew: How many crew are there?	1	2		
duration: What is the duration of the mission?	7 days	14 days		
habitat: How is a habitable volume provided by the system?	Habitat 1.7	Habitat 1.14	Habitat 2.14	Habitat 2.28
power: How is power provided by the system?	7-day battery	14-day battery		

Table 10. List of logical constraints for the example problem.

Constraint Name	Decision Variables Affected	Constraint Statement
Constraint i	# crew, duration	if (# crew = A), then (duration = A|B), elseif (#crew = B), then (duration = B)
Constraint ii	#crew, habitat	if (#crew = A), then (habitat = A|B|C|D), elseif (#crew = B), then (habitat = C|D)
Constraint iii	#crew, duration, habitat	if (#crew = A & duration = A), then (habitat = A|B|C|D), elseif (#crew = A & duration = B), then (habitat = B|C), elseif (#crew = B & duration = A), then (habitat = C|D), elseif (#crew = B & duration = B), then (habitat = D)
Constraint iv	duration, power	if (duration = A), then (power = A), elseif (duration = B), then (power = B)

This set of logical constraints illustrates all three examples of the types of logical constraints that exist. Constraint 1 is an example of a requirement constraint. While there is no physical reason why a crew of two is required for a longer duration, it is assumed that the stakeholders have told the architect that it is a ground rule for the system. Constraint 2 and Constraint 3 are examples for incompatibility constraints; the habitat selected must physically be able to support the parameters of the system. Constraint 4 is a combination of a constraint due to incompatibility (a 14-day mission MUST have a 14-day battery system) and a constraint due to sensibility (a 7-day mission WILL ONLY have a 7-day battery system). It is assumed (for simplification) that the power requirements of the life support system are independent of the number of crew. A 7-day mission is technically feasible if performed with a 14-day battery system, but it is assumed in this example that the architect and relevant stakeholders have agreed that there is no reason to include such a combination of decisions in the evaluation process since such architecture is of no interest.

Constraints 2 and 3 also illustrate the concept of redundant constraints. By examining the two statements, it can be seen that Constraint 3 makes Constraint 2 redundant. In other words if Constraint 3 is applied, then Constraint 2 will not eliminate any additional architectures in terms of their feasibility. Also, Constraint 4 makes the second if-then statement of Constraint 3 redundant. In typical methods for creating optimization simulations to explore the trade space of the system, it would make sense to remove the redundant constraints and statements from the simulation to improve computational efficiency; however, it is recommended that the system architect keep the constraints and statements in the model when doing decision-based modeling because the inclusion of the constraint impacts the perceived influence of the decisions. For this system, the decision of which habitat to use is affected by two constraints from two different lines of reasoning, even if they are logically redundant.

2.2.5 Iterations

Once the above-mentioned four tasks have been completed, the system is now fully represented as a decision-based system architecture model. However, it is recommended that the system architect perform several iterations of the tasks to refine the representation. Information gained from defining the property functions may show the architect that additional property variables are required to differentiate the architectures. This would be the case when the set of characteristics for two given architectures are identical. Also, the task of setting the logical constraints may allow refinement of the choice of decision alternatives to simplify the model. In the case of the example problem, Constraint 4 (based on the outside requirement) has eliminated the combination of parameters that would lead to a mission of 1 crew for 14 days. Therefore, Alternative B for the habitation element is no longer optimally designed for any of the system concepts and it should be apparent to the architect in this simple case that no 'optimal' architecture would exist based on the known set of property variables. In this case, the architect should discuss this issue with the relevant stakeholders and if they believe the alternative should be included in the model, then the architect should discuss the inclusion of other property variables which may make an architecture with such a habitat appear 'optimal' based on other criteria. In the example problem, it is assumed that the stakeholders agreed to the elimination of this design choice, thereby creating the final representation of the system as summarized by the following tables. Note the change in nomenclature for the alternative habitat designs.

Table 11. Final Decision variables for the example problem.

Decision Variables	Alt A	Alt B	Alt C
#crew: How many crew are there?	1	2	
duration: What is the duration of the mission?	7 days	14 days	
habitat: How is a habitable volume provided by the system?	Habitat 1.7	Habitat 2.14	Habitat 2.28
power: How is power provided by the system?	7-day battery	14-day battery	

Table 12. List of logical constraints for the example problem.

Constraint Name	Decision Variables Affected	Constraint Statement
Constraint i	# crew, duration	if (# crew = A), then (duration = A|B), elseif (#crew = B), then (duration = B)
Constraint ii	#crew, habitat	if (#crew = A), then (habitat = A|B|C), elseif (#crew = B), then (habitat = C)
Constraint iii	#crew, duration, habitat	if (#crew = A & duration = A), then (habitat = A|B|C), elseif (#crew = A & duration = B), then (habitat = B|C), elseif (#crew = B & duration = A), then (habitat = B|C), elseif (#crew = B & duration = B), then (habitat = C)
Constraint iv	duration, power	if (duration = A), then (power = A), elseif (duration = B), then (power = B)

Table 13. Property functions for the example problem.

Property Variable	Property Function
Mass of System	sum(element masses)
Number of crew-days	Number of crew x Mission duration

Table 14. Characteristics for the example problem decision alternatives.

Decision Variables	Alternatives	Characteristics		
		Number of Crew [-]	Mission Duration [days]	Element Mass [mt]
Number of crew	1	1	-	-
	2	2	-	-
Mission duration	7	-	7	-
	14	-	14	-
Habitation functionality	Habitat 1.7	-	-	1.3
	Habitat 2.14	-	-	2.6
	Habitat 2.28	-	-	2.9
Power provision functionality	7-day battery	-	-	2.0
	14-day battery	-	-	4.0

The representation can also be shown in graphical form as developed by the ADG framework, but for more complex systems with many decisions and constraints, these graphical formats can be difficult to comprehend and the tabular format has proven to present the data more succinctly. Several other tabular formats are feasible and are shown throughout the case studies. At this point, the representation can be turned into a computer simulation, however, knowledge can be gained about the system before simulation occurs which can assist in simulation as well as provide useful knowledge about the system.

2.3 Structural Reasoning

Structural reasoning is the second step in a decision-based system architecture model and it is used to transform the information from the representation created previously into a sorted set of decisions by reasoning about its structure. As discussed in the original ADG framework, the purpose of structural reasoning is two-fold: it is used to increase computational efficiency as well as to provide initial insight to the architect. As this research did not evolve the process or thinking related to the step of structural reasoning for decision-based modeling, the reader is directed to the research available on the original ADG framework for this step [Sim08].

Table 15 shows a tabular visualization of the example problem that assists in performing the structural reasoning. Each row represents a decision variable and each column represents either a decision variable or property variable. A box is marked in if there is a connection between the decisions through a logical constraint or a connection between a decision and a property variable through a property function. Table 16 shows the rankings of the four decisions based on their number of alternatives and their number of connections through the logical constraints.

Table 15. Tabular visualization of the representation of the example problem.

	#crew	duration	habitat	power	mass	crew-days
#crew	▨	C4	C2, C3			•
duration	C4	▨	C3	C1		•
habitat	C2, C3	C3	▨		•	
power		C1		▨	•	

Table 16. ADGsort1 and ADGsort2 rankings for example problem.

Decision	Degree of connectivity	Number of alternatives
#crew	3	2
duration	3	2
habitat	2	3
power	1	2

2.4 Simulating

The third step in creating a decision-based model is simulating. The simulating process transforms the structured decision problem into an executable computer model based on Constraint Satisfaction Problems (CSPs) and constraint network theory [RN02]. The goal of simulating is to enumerate the feasible combinations of decisions and their alternatives (i.e. feasible architectures) and to evaluate those feasible architectures based on the calculated property variables.

The simulating step requires two tasks to be completed. The first task is to create the simulation model based on the representation of the system. This involves choosing the specific strategy and tool that will be utilized to solve the value-based CSP that is created from the system representation. The original ADG framework used the Object-Process Network (OPN) [Koo05]. OPN is a meta-language that was created for the purpose of encoding, enumerating, and evaluating system architecture design spaces. Further discussion of OPN and its application for decision-based modeling can be found in the research related to the original ADG framework [Sim08]. For this research, several other strategies and tools were investigated for simulating the case studies.

One of the major discussion points focused on the choice between using a full-enumeration and partial-enumeration strategy for solving the value-based CSP. Full-enumeration strategies create simulations that enumerate and evaluate all feasible combinations of decisions regardless of their optimality. Partial-enumeration strategies, such as guided improvement algorithm [REJ09] or other heuristic algorithms do not enumerate all feasible combinations but have a high possibility of providing the set of algorithms that are considered 'optimal' based on the given property variables known as architectures that are on the "Pareto Front" (i.e. there exist no other solutions that can improve any given property variable without negatively impacting another). Strategies and tools have also been developed to investigate the enumeration of a set of architectures that exist along a "fuzzy Pareto front" (i.e. are within a set percentage of the optimal architectures). The benefit of partial enumeration strategies are that they allow improved computational efficiency by disregarding feasible, but dominated (i.e. not optimal) architectures. Full enumeration and evaluation of the entire feasible set typically requires more computation time and effort compared to partial enumeration strategies. The benefit of full enumeration and evaluation of the entire feasible set is that it provides more information to the system architect to utilize in the investigation of the apparent influence of the related decisions, as will be shown in the next section. This benefit was the reason full enumeration and evaluation was chosen for the case studies completed in this research.

The strategy of full enumeration and evaluation of the feasible architectures was implemented in the software package known as MatLab. This software was chosen because of its ability to implement all types of property functions (including non-separable functions) and the relative ease of programming the problems in MatLab. The ease of use of programming with the software is paramount in creating decision-based system architecture models since in most cases the simulation require several iterations and refinements. MatLab had the additional benefit of being able to be used for creating the views necessary for analyzing the data created by the simulation.

The second part is the execution of the model to produce the set of feasible combinations of decisions or architectures. Executing the relevant MatLab code for the system of interest is all that is required. The code for the example problem simulation is shown in Appendix A. The simulation of the example problem produced 11 feasible archtiectures from 24 unconstrained combinations of decisions.

2.5 Viewing

The viewing process transforms the feasible combinations of decisions and their property variables into new available knowledge. In order for this information to be useful in the decision-making process, it must be presented in a way that is meaningful to the architect and the stakeholders. The overall goal of viewing the simulation data is to improve the architect's ability to comprehend the space of feasible architectures and gain insight into the related decisions.

Similar to the original ADG framework, which provided two ways of viewing the information provided by a simulation (the Pareto Front View and the Decision Space

View), this research presents two types of information that can be taken from a decision-based system architecture model, both of which have been evolved through this research:

- Information related to investigating the "optimal" architectures based on the property variables of interest. This information is presented through an extension of ADG's concept of Pareto Front Views.
- Information related to the influence of each individual decision. The DSV concept from the ADG framework is evolved to enable information related to the system as a whole, not just for each single property variable.

The following subsections discuss each of these types of information and their related views respectively.

2.5.1 Identifying the preferred architectures

The original ADG framework provided Pareto Front Views as a way to view the information related to identifying the preferred architectures. The Pareto Front View allows two property variables to be plotted against each other to show the non-dominated architectures. Figure 6 shows a plot of the feasible architectures based on their calculated system masses and number of crew-days supported. The 'utopia point' on the chart is in the lower right corner of the chart and represents the most crew-days supported for the lowest system mass. The Pareto Front for this plot shows that there are three non-dominated architectures (one for each possible value of crew-days supported). The three non-dominated architectures are the three architectures that have the habitat and power system ideally designed for their specific system parameters.

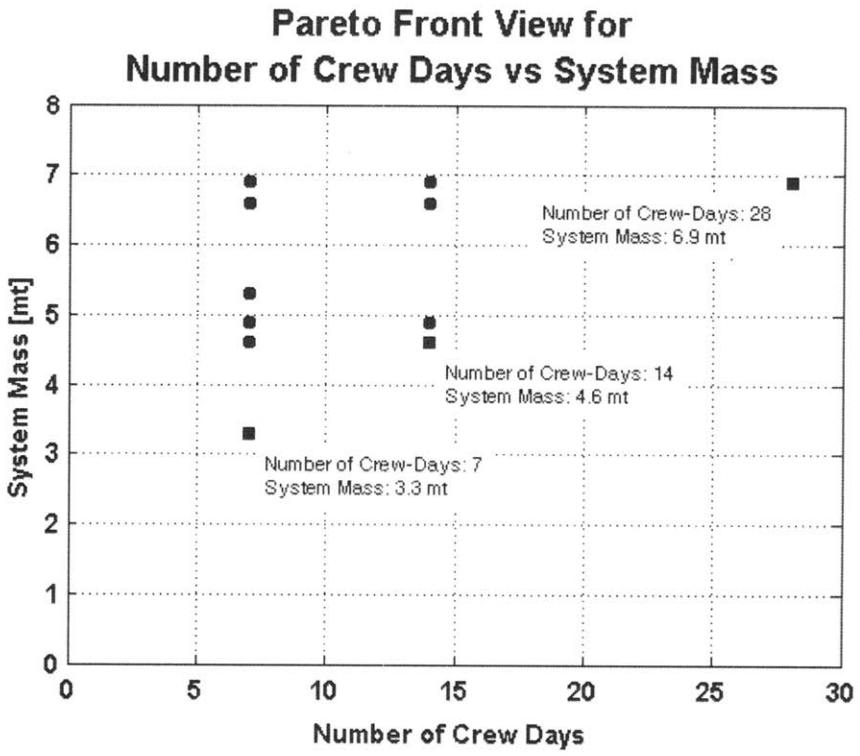


Figure 6. Pareto Front View for the example problem.

Table 17. Values for the dominant architectures for the example problem.

Arch #	Number of Crew	Mission Duration [days]	Habitat	Power System	Crew-days supported	System Mass [mt]
1	1	7	Habitat 1.7	7day battery	7	3.3
7	2	7	Habitat 2.14	7day battery	14	4.6
11	2	14	Habitat 2.28	14day battery	28	6.9

While Pareto Front Views work well in comparing two property variables against each other, they are challenged to express useful information for systems with multiple property variables. Complex systems, such as human spaceflight projects (which have not been simplified), tend to have at least half a dozen property variables, if not many more. In these cases, Pareto Front Views are not the best option for evaluating preferred architectures. There are multiple methods for viewing the information related to evaluating complex systems. One of the options used in this research was the application of Multi-Attribute Utility Theory (MAUT) [Dye92].

This allows architectures with many property variables to be evaluated against each other using a single number to represent the benefit of the architecture. Using weighting factors to show the importance of each property variables based on stakeholder inputs, multiple property variables can be combined to allow a single overall metric to represent the benefit of the system. Using this metric, the ratio of system benefit versus cost can be used to evaluate each potential architecture.

MAUT also allows the possibility of setting cut-off values for property variables and the possibility of representing non-linear scaling effects with property variables due to effects such as diminishing returns. For example, in the system discussed above, the actual benefits accrued from increasing crew-days related to the amount of science knowledge that is gained may not continuously increase. Assuming that all the relevant scientific experiments can be accomplished within 14 crew-days, then there is no actual benefit gained from increasing the crew-days to 28.

Using MAUT, a representative value can be given to the property variables to account for this cut-off of benefit. It is assumed that zero crew-days produce zero benefit and 14 crew-days produce 100% of benefits. Using this benefit metric changes the ratio of apparent benefits to costs (in this case mass) of the system as can be seen in Table 18. Through use of the property variables alone, it can be seen that Arch #11 appears to have the best value (benefit over cost ratio), however through the use of MAUT and the use of a non-linear benefit metric for crew-days, it can be seen that the actual best value is in Arch #7.

Table 18. Benefit/Cost ratios for example problem.

Arch #	System Mass [mt]	Crew-days supported	Crew-Days/Mass	Benefit Metric	Benefit/Mass
1	3.3	7	2.1	50	15.2
7	4.6	14	3.0	100	**21.7**
11	6.9	28	**4.0**	100	14.5

More discussion of the application of MAUT to decision-based system architecture modeling can be seen in the next chapter covering the LSS case study.

2.5.2 Determining the influence of system architecture decisions

The original ADG framework developed the Decision Space View (DSV) to allow system architects to determine high-influence decisions. The DSV plots each of the decisions in a two-dimensional space consisting of a measurement of their connectivity on the horizontal axis and a measurement of the sensitivity of a given property variable to the decision in question. High-influence decisions are the decisions that strongly affect the feasible set of alternatives for other decisions (i.e. are highly connected to other decisions) and strongly influence the properties of the system.

Figure 7 is an overview of the Decision Space View. In Figure 7, the horizontal axis, the degree of connectivity, is a measure of how many other decision variables are connected to a particular decision through the logical constraints. This can be considered a first order measure of the impact of one variable on the feasible set of the other decisions. The vertical axis, the Property Value Sensitivity, is a measure of how much influence a decision has on system metrics.

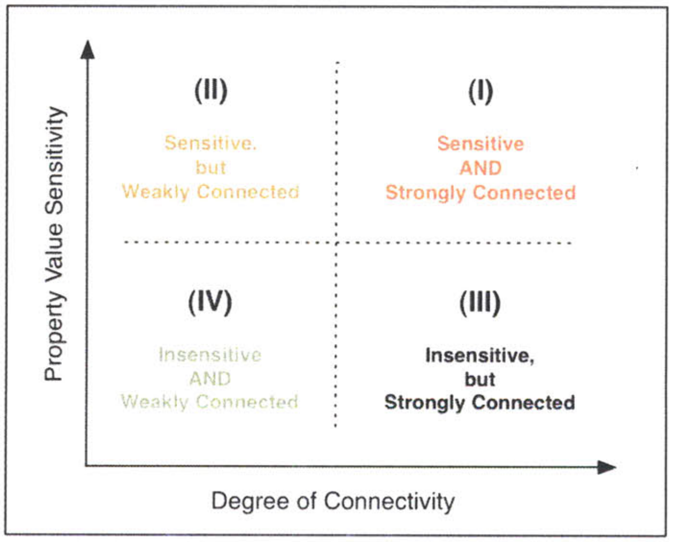

Property Value Sensitivity
(II)
Sensitive, but Weakly Connected
(I)
Sensitive AND Strongly Connected
(IV)
Insensitive AND Weakly Connected
(III)
Insensitive, but Strongly Connected
Degree of Connectivity

Figure 7. Decision Space View indicating the partitioning of decisions by Property Value Sensitivity and Connectivity [Sim08].

To aid in interpreting the Decision Space View, it is split into four quadrants, representing:

- Sensitive and strongly connected decisions: strong influence in both metrics.
- Sensitive, but weakly connected decisions: These influence performance, but do not impact many other decisions.
- Insensitive, but strongly connected decisions: These decisions influence the choices available for other decisions, but do not strongly influence metrics.
- Insensitive and weakly connected decisions. These decisions have little affect on metrics and the feasible space of other decisions.

As shown in Equation 1, the Property Value Sensitivity (PVS) metric is calculated for each property and each decision over the set of feasible combinations of decisions and their associated properties. PVS is a measure of the average magnitude of change in a property that occurs when changing the assignment of a particular decision variable. Further background into the PVS can be found in the original research related to the ADG framework [Sim08].

Equation 1. The Property Value Sensitivity metric.

$$PVS_{m_j, d_k} = \frac{\sum_{a_i \in F} \left| E(m_j) - E(m_j \mid d_k = a_i) \right|}{\left| a_i \in F \right|}$$

where

PVS_{m_j, d_k} : Property Value Sensitivity for property m_j to change in decision d_k

$E(m_j)$: Mean value of property m_j over all members of the feasible set

$E(m_j \mid d_k = a_i)$: Mean value of property m_j over all members of the feasible set where decision d_k is set toa particular alternative a_i

$a_i \in F$: Number of alternatives a_i for decision d_k in the feasible set

While the DSV is an invaluable tool for viewing the information created by a decision-based model, there are two evolutions from the original ADG framework which should enhance the information given by the DSV: the ability to select the set of architectures included in the calculation of the DSV metrics and the ability to view the overall influence of a decision on the entire system as opposed to the influence on individual property variables.

One evolution of the decision-viewing tools for decision-based system architecture modeling is that the PSV, as originally devised by Simmons, looked at the sensitivity of the property variable over the entire set of feasible decisions. Through application to the case studies and related discussions, it was decided that a system architect may not be interested in how a decision impacts a property variable across the entire set of feasible designs including any fully dominated architectures. The system architect may want to focus on the set of architectures that are on the Pareto Front (i.e. non-dominated designs) or the system architect may want to focus on the set of architectures within a certain percentage of the Pareto Front. There may also be value in comparing the DSVs for the full feasible set against the DSVs for only the Pareto Front set. This may provide insight into whether the apparent influence of a decision when examining the entire feasible set is also present when only looking at dominant or interesting architectures.

It is recommended that the simulation strategy chosen for decision-based modeling uses full enumeration as opposed to partial enumeration. Figure 8 and Figure 9 show the DSV for the property variable of mass for the entire feasible set and for the non-dominated set of architectures, respectively. If the influence of the decisions were to be investigated based on the entire feasible set, the decision of setting the duration is not only the most connected decision, but also the decision to which system mass is most sensitive to. However, if only the non-dominated set is examined, it can be seen that system mass is just as sensitive to the choice of the power system as it is to the duration.

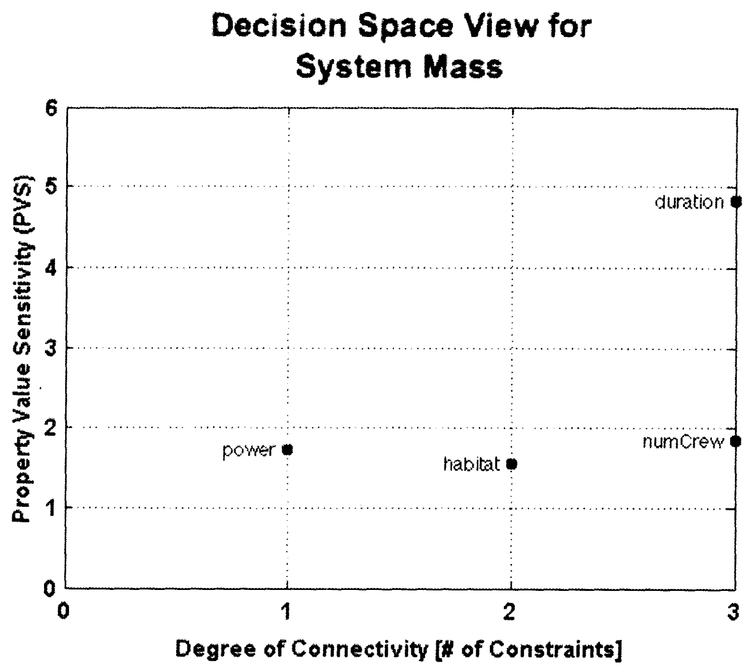


Figure 8. Decision Space View for mass for the entire feasible set of the example problem.

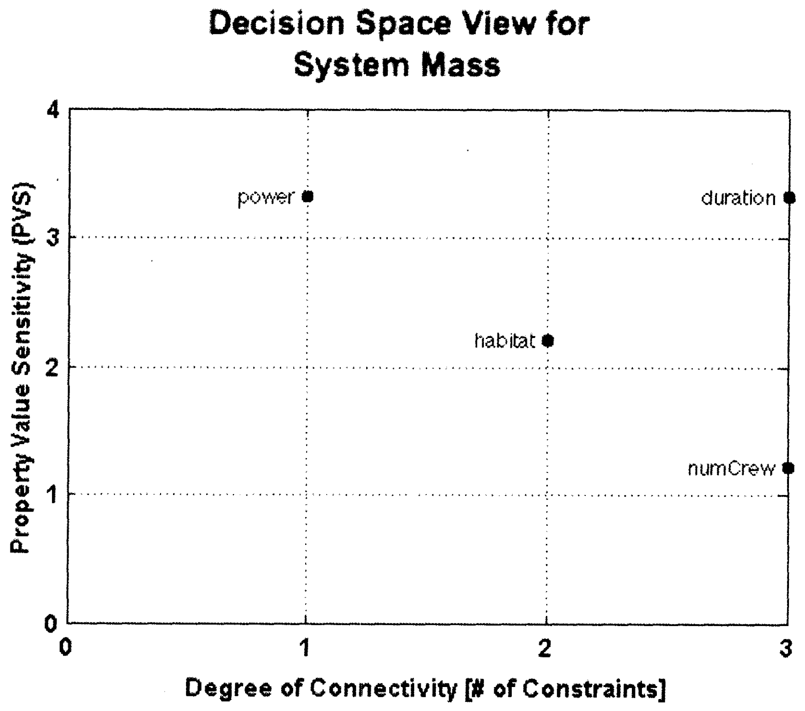


Figure 9. Decision Space View for mass for the non-dominated set for the example problem.

Decision Space Views can be developed for each property variables or metric of interest. The challenge is that for a large set of metrics, there is a large set of DSVs and related information that must be digested. In addition to this, the initial formulation of the DSVs did not allow a quantification of the influence beyond the above discussion of the four quadrants. To address these limitations, the DSV process was combined with a scoring system, known as the Average Influence Score, akin to the Risk Matrix [NASA, 2007] as shown in Figure 10. This allowed a calculation of a quantitative score for the overall influence of the decisions on the system's overall benefit and the systems overall cost. In addition, an AIS can be calculated for the overall value of the system by taking the average of the Benefit AIS and the Cost AIS.

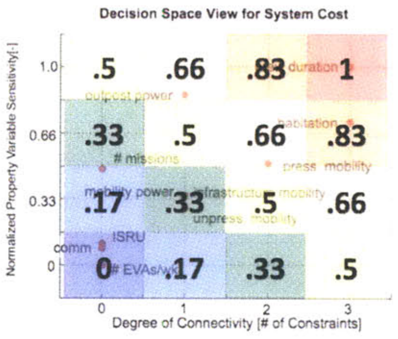


Figure 10. Grading for the Average Influence Score.

As opposed to using discrete values to calculate the Average Influence Score (AIS), as is done with typical risk matrices, the AIS values for the benefit property variables and for the cost related variables are calculated for a given decision using Equation 2. For the example problem, the calculation for the AIS for each decision is shown in Table 19. Note that the Property Value Sensitivity values used are for the DSVs taken for the non-dominated set.

Equation 2. The Average Influence Score.

(An AIS is developed for the set of benefit property variables and a second AIS is created for the set of cost property variables.)

$$\mathrm{AIS}_{d_j} = \frac{\sum_{m_i \in F} \left(0.5 \times \frac{PVS_{m_j,d_k}}{\max_{d_i \in F}(PVS_{m_j})} + 0.5 \times \frac{connectivity_{d_k}}{\max_{d_i \in F}(connectivity)}\right.}{\left|m_i \in F\right|}$$

AIS_{d_j} : Average Influence Score for decision j.

PVS_{m_j,d_k} : Property Value Sensitivity of decision d_k for property variable m_j.

$\max_{d_i \in F}(PVS_{m_j})$: Maximum value of PVS for property variable m_j for all decisions.

$connectivity_{d_k}$: Degree of connectivity of decision d_k.

$\max_{d_i \in F}(connectivity)$: Maximum degree of connectivity for all decisions.

$\left|m_i \in F\right|$: Number of property variables.

Table 19. AIS calculation for the example problem.

	# of crew-days		Benefit AIS	Mass [mt]		Cost AIS	Value AIS
	Connectivity	PVS		Connectivity	PVS		
numCrew	3	7	0.66	3	1.23	0.69	0.67
duration	3	21.43	1.00	3	3.32	1.00	1.00
habitat	2	13.23	0.64	2	2.22	0.67	0.65
power	1	21.43	0.67	1	3.32	0.67	0.67

Using the Decision Space Views and the Average Influence Score, it can be seen that the most influential decision for the example problem is the choice of duration for the mission.

With the evolutions to both the Pareto Front Views and the Decision Space Views discussed in the previous section, a system architect can now use decision-based system architecture modeling to determine the architecture that delivers the most overall value and the decision that is most influential across the system.

2.6 Summary

This chapter presented the theory behind single-level decision-based system architecture modeling. In order to use a decision-based modeling framework such as

ADG, a system architect must iteratively go through the four steps of the framework as shown in Table 2. Each step of the process has two or more tasks that must be undertaken to complete that step. Each cycle through the framework produces new available knowledge allowing the system architect to refine their model. Major evolutions of single-level decision-based system architecting as compared to the ADG framework include:

- Guidelines for the decomposition of high-level stakeholder needs into a coherent set of quantifiable objectives to which the property variables can be mapped.
- Guidelines to ensure that the decision variables exist on a single level of abstraction of the architecture and that they are phrased to examine the attributes of the system and the form-to-function mapping required for the system.
- Classifications for logical constraints in terms of their objectivity or subjectivity.
- The incorporation of MAUT to allow the investigation of preferred architectures with more than two property variables and to account for property variables that do not scale linearly.
- The introduction of the Average Influence Score (AIS) to investigate the overall influence of a decision across all benefit and cost property variables.

Sections 2.2 through 2.5 discussed the theory behind each step of the process as well as the evolutions made by this research as compared to the original ADG framework. Each step of the process was illustrated through the application of the theory to a simplified example human spaceflight project. The next two chapters each cover two single-level decision-based modeling case studies that were investigated as part of this research. Chapter Three covers the investigation of the Lunar Surface System architecture and Chapter Four covers the investigation of the architectures of deep space missions. After these case studies have been discussed, this thesis turns it attention to developing a multi-level decision-based modeling framework. Chapter Five discusses the theory using an extension of the case study shown here. Chapter Six applies the multi-level framework to further investigating deep space missions. All of the research is summarized in Chapter Seven that also includes recommendations for future work.

3 The Lunar Surface System: A Case Study in Single-level Decision-Based System Architecting

3.1 Introduction

On January 14, 2004, then President George W. Bush announced the Vision for Space Exploration [Bus04]. As part of this new space policy, NASA began investigating how humans might one day live and work on the lunar surface. A major part of this effort is the architecting of the Lunar Surface System (LSS); the system that consists of all lunar surface assets (e.g. habitation, power, mobility, etc.) required to support humans on the Moon for extended periods.

The process of architecting the LSS is interesting to investigate because it is a challenging task. This is because the system itself is highly complex and has both unknown preferred functional allocations and unknown preferred operational strategies. On top of this, the system has a low level of "market knowledge" meaning that there is no clear understanding of how to gauge the ultimate goals of the system and how it ultimately provides value. Because of all these reasons, architecting the Lunar Surface System is an unprecedented design problem and an excellent case study for investigating the application of decision-based system architecture modeling.

The overall objective of this case study is to comprehensively explore the architecture trade space of the LSS and identify preferred architectures while gaining insight into the influential decisions by modeling the system as a set of interconnected system architecture decisions using a decision-based system architecture modeling.

This is not the first time that NASA and other organizations have investigated the architecture of a lunar outpost or the Lunar Surface System. In the late eighties, NASA investigated several possible architectures during the 1988 and 1989 Case Studies [NAS88, NAS89]. In 1993, NASA proposed an alternative architecture known as the First Lunar Outpost [NAS93] and again they looked at another architecture in 1996 that focused on the use of early lunar resource utilization [NAS96]. Several other groups have also proposed different variations of architecture for a lunar outpost in the past [Men85, Eck07, Hof07].

More recently, since the announcement of the Vision for Space Exploration in 2004, NASA has undertaken several cycles of investigation into the system architecture for the LSS. In 2005, NASA undertook the Exploration Systems Architecture Study [NAS05], which discussed the architecture of a lunar outpost in the context of how its system would impact the overall lunar transportation system (i.e. launch vehicles, in-space capsules, and lunar landers). In 2007, NASA began the first of the Lunar Architecture Team (LAT) studies. The result from the LAT1 studies was a single baseline design for a LSS that could support up to four crew for missions of up to 180 days [NAS07]. In 2008, NASA expanded on the LAT1 baseline by investigating six variations of the proposed outpost during the LAT2 cycle. The findings from this cycle were later expanded upon during the 2009 Constellation Architecture Team (CxAT) studies [NAS09]. These studies investigated 13 different scenarios for the LSS in terms of what elements and functions the system would consist of.

NASA's current approach to investigating the architecture of the LSS is to develop a limited set of detailed "lunar scenarios" [NAS09]. Each new "scenario" is based on knowledge gained from previous "scenarios". This approach has led to an incomplete investigation of the entire LSS trade space for several reasons. Since the new "scenarios" are based on previous "scenarios", they tend to be evolutionary in nature by building on or changing a single aspect of the previous design. This eliminates the possibility of investigating revolutionary ideas. On top of this, the approach leads to team members championing certain concepts and elements that impact the evolution of future 'scenarios'. Because of these facts, NASA would benefit from a comprehensive investigation of the overall architecture trade space as well as an investigation into how each decision affects the overall system and the value it delivers, which is what this case study attempts. Specifically, this case study has two primary objectives. These objectives are:

1. To enumerate and evaluate the feasible architectures for the Lunar Surface System.
2. To investigate the influential the decisions for the Lunar Surface System.

The rest of this chapter attempts to discuss and detail the steps, as discussed in the previous chapter, required to apply single-level decision-based system architecture modeling to the investigation of the LSS. Section 3.2 focuses on properly representing the problem as a decision-based system architecture model. Section 3.3 discusses how insight can be gained about the system by reasoning about the structure of the problem. Section 3.4 discusses how the represented system can be simulated to enumerate and evaluate the feasible architectures. Section 3.5 focuses on viewing the information about the architectures and the decisions in the model. Finally, section 3.6 wraps up the chapter by summarizing the insights about the LSS.

3.2 Representing

In order to effectively represent a system architecting problem in a decision-based model, all four types of data must be developed that were discussed in the previous chapter including: property variables, decision variables, property functions, and logical constraints. As mentioned previously, there are four key tasks be undertaken in order for this data to be represented:

1. The property variables for the system must be selected.
2. The set of decision variables to be considered must be created including the possible alternatives for each decision.
3. The property functions must be formulated to link the decision variables to the property variables.
4. The logical constraints, which limit the possible assignments for the decision variables, must be enumerated to ensure only feasible architectures are enumerated.

The following subsections will go through each of these specific tasks in terms of the application to the LSS.

3.2.1 Choosing the Property Variables

The property variables for the decision-based model of the LSS need to capture both the cost and the benefit of the overall system to NASA. Determining a property variable for cost is perhaps simpler then determining property variables for the benefit of the system because there is less ambiguity in what the cost of the system entails compared to what the benefit of the system is.

In any space mission, the total mass of the system is considered a good first-order approximation for the cost [Wer99], however, instead of using purely a mass-based property variable, further definition of the overall cost of a feasible LSS architecture can be achieved by using mass-based cost models for the development and production of system elements and estimates for the cost of transporting theses elements from Earth to the Lunar Surface. Resources are available from NASA to estimate the production and development of certain space elements based on parametric modeling tools [JSC09]. Previous research has provided reliable estimates for the life-cycle cost of both crewed and un-crewed lunar transportation systems for delivering both crew and cargo to the Moon [Hof99].

Using these tools, it is possible to determine an estimate of the overall system life-cycle cost of the LSS including both element development and production as well as transportation costs to the lunar surface and a simplistic mass-based lifecycle cost model was able to be used to evaluating the feasible LSS architectures. It should be noted that it is not the actual specific cost values for each architecture that is the most important; instead it is the relative comparison between the architectures based on the cost numbers that is important.

The real challenge of choosing the property variables comes in determining a comprehensive and objective set to capture the benefit of the LSS. As discussed in the previous chapter, the most efficient means to accomplishing this is by understanding the high-level benefits of the system and decomposing them into goals and objectives and quantified benefit indicators or property variables. NASA has presented a set of six "exploration themes" that are meant to capture the overall benefit of exploring the lunar surface as shown in Table 20 [Coo08]. While these themes are comprehensive in terms of capturing the overall high-level benefits of the LSS, based on the descriptions, it is easy to see the difficulty in objectively and quantitatively comparing feasible system architectures for the LSS based on these themes directly. For example, it is difficult to quantify the direct change in benefit to "exploration preparation" depending on whether a feasible architecture has the capability to support 1 EVA event a week or 3 EVA events a week. It is this reason that it is necessary to formally decompose these themes to a set of more quantifiable properties that can be used for comparison.

Table 20. NASA's lunar exploration themes and their descriptions [Coo08].

Exploration Themes	Theme Descriptions
Exploration Preparation	Reduce the risks and increase the productivity of future missions in our solar system by testing technologies, systems, and operations in an off-Earth planetary environment.
Scientific Knowledge	Engage in scientific investigations of the Moon, on the Moon, and from the Moon.
Human Civilization	Develop the knowledge, capabilities, and infrastructure required to live and work on the Moon, with a focus on continually increasing the number of individuals that can be supported on the Moon, the duration of time that individuals can remain on the Moon, and the level of self-sufficiency of lunar operations.
Economic Expansion	Create new markets, based on lunar and cis-lunar activity, that will return economic, technological, and quality-of-life benefits to all humankind.
Global Partnerships	Enhance global security by providing a challenging, shared, and peaceful global vision that unites nations in collaborative pursuit of common objectives.
Public Engagement	Use a vibrant exploration program to excite the public about space, encourage students to pursue careers in high technology fields, and ensure that individuals enter the workforce with the scientific and technical knowledge necessary to sustain exploration.

Using the process described in the previous chapter, each of the themes were decomposed into a limited, but still comprehensive, set of more specific goals, which, in turn, were decomposed further into a set of detailed objectives (See Appendix E). These objectives were developed at such a level that quantitative benefit indicators or property variables could be used to determine the completion of each objective and could then be used to evaluate each feasible LSS architecture.

The set of 16 property variables chosen are shown in Table 21 and Table 22 with their name, units, description, and relationship to the benefit themes.

Table 21. **Property Variables for the LSS decision-based model.**

Property Variable	Description	Exploration Themes Impacted
Number of astronauts [#]	The total number of astronauts landed on the lunar surface during a phase of lunar exploration.	Exploration Preparation, Human Civilization, Economic Expansion, International Partnerships, Public Engagement
Number of 60+ Day Missions [#]	The total number of missions during a phase of lunar exploration that had a duration of 60 days or greater.	Scientific Knowledge, Human Civilization
Total Crew Time [# hrs]	The total amount of crew time spent on the lunar surface.	Exploration Preparation, Scientific Knowledge, Human Civilization, Economic Expansion, International Partnerships, Public Engagement
Extra-Vehicular Activity (EVA) Time [# hrs]	The total amount of crew time dedication to extra-vehicular activities.	Exploration Preparation, Scientific Knowledge, Human Civilization
Intra-Vehicular Activity (IVA) Crew Time [# hrs]	The total amount of crew time dedicated to activities inside the lunar habitats.	Exploration Preparation, Scientific Knowledge, Human Civilization
Exploration Range Enabled [km]	The maximum exploration range achievable by the LSS.	Scientific Knowledge
Pressurized Volume [m3]	The total amount of pressurized volume provided by the LSS.	Scientific Knowledge, Public Engagement
Communications Capability [-]	A binary metric related to the presence of augmented (high-bandwidth) communication systems on the lunar surface.	Public Engagement

Table 22. Property Variables for the LSS decision-based model (continued).

Property Variable	Description	Exploration Themes Impacted
Experience with Environmental Control and Life Support Systems (ECLSS) [days]	The total number of days of experience gained with Environmental Control and Life Support Systems (ECLSS).	Exploration Preparation, Human Civilization
Experience with Pressurized Rover Operations [# days]	The total number of days of experience gained in operating pressurized rovers on the lunar surface.	Exploration Preparation, Human Civilization
Experience with In-Situ Resource Utilization (ISRU) Systems [# days]	The total number of days of experience gained in operating In-Situ Resource Utilization (ISRU) systems on the lunar surface.	Exploration Preparation, Human Civilization
Experience with Power Systems [# days]	The total number of days of experience gained in operating outpost power systems on the lunar surface.	Exploration Preparation, Human Civilization
Experience in Dust Mitigation [# of EVA events]	The total number of events from which experience in dust mitigation can be gained (i.e. EVA events).	Exploration Preparation, Human Civilization
Number of Cargo Flights [#]	The total number of automated cargo flights undertaken for a single phase of lunar exploration.	Exploration Preparation, Human Civilization
Number of Crewed Flights [#]	The total number of crewed flights to the lunar surface undertaken for a single phase of lunar exploration.	Exploration Preparation, Human Civilization
Utilization Mass Available [mt]	The total unused mass capability of the transportation system that can be dedicated to useful payloads (e.g. science packages or public engagement tools).	Scientific Knowledge, Economic Expansion, International Partnerships, Public Engagement

Once the property variables have been determined, it is possible to analyze each property variable and decompose them to determine what information is required to calculate each one. This process allows for a derivation of a rough list of decision variables and a starting point for the next step in representing the system: formulating the decisions.

3.2.2 Formulating the Decisions

Using an understanding of the property variables as a starting point to determine what decisions should be included, the decision list was developed by comparing and contrasting the features of past and present proposed architectures in the literature that were discussed at the beginning of this chapter to ensure that the list captured all relevant trades and options.

Based on the review of previous studies and the guidelines provided for this case study, the overall concept for the LSS used in this case study was framed as follows:

- The LSS architecture would support crews exploring a single polar location of the lunar surface.
- The LSS architecture would consist of a set of cargo flights (the amount set by the mass of the assets delivered) and a set of crewed missions (the amount to be traded)
- The crewed missions will all be assumed to be of the same duration of a value which would be traded
- The LSS architecture would support a set number of EVAs per week (the amount to be traded)

These guidelines along with the set of property variables selected led to the development of three decisions related to setting the attributes of the system. These decisions include:

- How many crewed missions will occur?
- What is the duration for each crewed mission?
- How many EVAs per crewmember per week?

As discussed in the previous chapter, along with the decisions related to setting the attributes, there is a class of decisions related to mapping form to function for the system. Based on review of previous and current studies, the internal functionality of the LSS was considered to include some or all of the following:

- Transporting large assets, such as habitats, on the surface to assist in exploration and any necessary assembly.
- Providing mobile pressurized exploration capability for the crew.
- Providing mobile unpressurized exploration capability for the crew.
- Providing power sources for the mobile pressurized vehicles, if present.
- Providing habitation for the crew while on the lunar surface.
- Providing power to this habitation on the lunar surface.
- Providing augmented communication capability to support high bandwidths.
- Providing the capability for in-situ production of oxygen.

A decision variable was created to address what form would be mapped to each of these internal functions and each decision was populated with applicable elements designed for other studies to address these functions. This led to a list of 11 decisions to be included in the model as shown in the morphological matrix presented in Table 23. The alternatives were of feasible forms were populated through review of the elements created for previous LSS-related studies and were chosen to allow a comprehensive representation of the entire trade space available.

Table 23. Morphological matrix of the decisions included in the LSS model.

Decision Variable	Alt A	Alt B	Alt C	Alt D	Alt E
How many crewed missions will occur?	1	3	5		
What is the duration for each crewed mission?	7	14	28	60	180
How many EVAs per crewmember per week?	1	2	3		
How is large element mobility provided?	none	1x heavy lift mobility	2x heavy lift mobility		
How is pressurized crew mobility provided?	none	2x pressurized rover	4x pressurized rovers		
How is unpressurized crew mobility provided?	none	1x small unpressurized	2x small unpressurized		
How is mobile power provided?	none	Solar array w energy storage	Radioisotope Power Source		
How is habitation for the crew provided?	none	28d single module	60d single module	180d dual module (assembled)	180d dual module (not assembled)
How is outpost power provided?	none	Solar array w energy storage	Fission Surface Power		
How is augmented communications provided?	none	1x Communication Terminal			
How are in-situ resource utilization capabilities provided?	none	2x Oxygen Production Plants			

3.2.3 Developing the Property Functions

The third step of representing a system is to develop the property functions that connect the decisions (and their alternatives) to the property variables selected for evaluating the architectures. As discussed in the previous chapter, the property functions require:

1. The formula required for calculating the metric.
2. The characteristics of the decision alternatives that are the values for input into the formulas.

The property functions were derived by analyzing each of the property variables and determining what information related to the decisions, as well as any other assumptions, were required. Table 24 shows the property functions for each of the 17 property variables included in the model. In addition to information gathered from the decisions and their alternatives, certain assumptions are required including:

- For each crewed mission, there are exactly four astronauts.
- Each EVA duration is exactly 8 hrs.
- Any time not spent on EVA is considered as IVA time.
- Each cargo flight has a capacity of 14.5 mt for cargo.
- Each crewed flight has a capacity of 1 mt for cargo.
- Each crew-member requires 10 kg of supplies per day.

Table 24. A list of the metric functions included in the LSS model.

Property Variable	Property Function
numAstronauts	numCrewedMissions x 4crew

num60dayMissions	if (duration >= 60), then numCrewedMissions
	else 0

totalCrewTime [hr]	numCrewedMissions x duration x 4crew x 24 hours a day
EVACrewTime [hr]	numCrewedMissions x duration x numEVAperWeek/7 x 4crew x 8hr
IVACrewTime [hr]	totalCrewTime – EVAcrewTime
explorationRange [km]	max(range of infrastructure mobility, range of pressurized mobility, range of unpressurized mobility)
pressurizedVolume [m^3]	sum(volume of pressurized mobility, volume of habitats)
communications	1 = enhanced comm., 0 = no enhanced comm.
ECLSSexp [days]	if (60day or greater habitat is present), then (numCrewedMissions x CrewDuration) else (0)
roverOpsExp [days]	if (pressurized mobility is present), then (numCrewedMissions x CrewDuration) else (0)

ISRUexp [days]	if (ISRU is present),
	then (numCrewedMissions x CrewDuration)
	else (0)

powerExp [days]	if (outpost power is present), then (numCrewedMissions x CrewDuration) else (0)
dustMitigationExp [events]	numCrewedMissions x CrewDuration x numEVAperWeek/7
numCargoFlights	(sum(element masses)+numCrewedMission x CrewDuration x 10)/14500
numCrewedFlights	numCrewedMissions
utilizationMass	(numCargoFlights x 14500) + (numCrewedFlights x 1000) – (sum(element masses)+numCrewedMission x CrewDuration x 10)
lifeCycleCost	function(element choices, numCargoFlights, numCrewedFlights)

In order to calculate these property functions, information is required about the alternatives of each decision variable. This is known as the characteristics of each alternative. As shown by the list of property functions, it can be seen that the following characteristics (beyond the apparent characteristics of duration, number of crewed missions, and EVAs per week) are required:

- Exploration range enabled by mobility assets
- Volume of pressurized assets
- Mass of element alternatives
- Cost of element alternatives

Due to the increased number of decisions and alternatives as compared to the example problem in the previous chapter, the characteristics will not be shown in a single chart. Alternatively each of these characteristics is shown below in its own table.

Table 25. Characteristic exploration range for the mobility elements of the LSS.

EXPLOATION RANGE CAPABILITY [km]					
DV Short Name	**Alt A**	**Alt B**	**Alt C**	**Alt D**	**Alt E**
numCrewMissions					
crewDuration					
numEVA					
infrastructureMobility	0	1000	1000		
pressMobility	0	100	100		
unpressMobility	0	10	40		
mobilePower					
habitation					
outpostPower					
comm					
isru					

Table 26. Characteristic volumes for the pressurized elements of the LSS.

ELEMENT VOLUME ESTIMATES [mt]					
DV Short Name	**Alt A**	**Alt B**	**Alt C**	**Alt D**	**Alt E**
numCrewMissions					
crewDuration					
numEVA					
infrastructureMobility					
pressMobility	0	26	52		
unpressMobility					
mobilePower					
habitation	0	55	110	165	165
outpostPower					
comm					
isru					

Table 27. Characteristic masses for each element of the LSS.

ELEMENT MASS ESTIMATES [mt]					
DV Short Name	**Alt A**	**Alt B**	**Alt C**	**Alt D**	**Alt E**
numCrewMissions					
crewDuration					
numEVA					
infrastructureMobility	0	2.3	4.7		
pressMobility	0	8.3	16.7		
unpressMobility	0	0.23	0.46		
mobilePower	0	1.3	5.5		
habitation	0	7.5	11.2	13.3	13.3
outpostPower	0	14.5	8.7		
comm	0	0.58			
isru	0	0.84			

Table 28. Characteristic cost for R&D of each element.

ELEMENT R&D COST ESTIMATES [M$ FY04 USD]					
DV Short Name	**Alt A**	**Alt B**	**Alt C**	**Alt D**	**Alt E**
numCrewMissions					
crewDuration					
numEVA					
infrastructureMobility	0	940	1100		
pressMobility	0	2000	2500		
unpressMobility	0	430	650		
mobilePower	0	630	1300		
habitation	0	1700	2000	2800	2800
outpostPower	0	2000	5100		
comm	0	580			
isru	0	840			

With the property variables, decision variables, and property functions developed, the system representation can now enumerate and evaluate all possible architectures, but there is nothing to eliminate infeasible architectures from the evaluations. In order to eliminate the infeasible architectures from being enumerated, the logical constraints must be enumerated.

3.2.4 Enumerating the Constraints

Following the guidelines on the three types of logical constraints discussed in the previous chapter, there are eight constraints contained within the LSS model. Each of these constraints can be seen below:

- Constraint #1: Mobile power must be present when pressurized mobility is present and if no pressurized mobility is present, no mobile power will be present (incompatibility and sensibility)
- Constraint #2: The choice of habitation must be in-line with the choice of crew duration and vice-versa. (i.e. the habitation must be large enough to support a given crew duration, but no larger) (incompatibility and sensibility)
- Constraint #3: If the habitation chosen requires assembly, then there must be some sort of infrastructure mobility (incompatibility)
- Constraint #4: If the crew duration is only 7 days, there is no pressurized mobility. If the crew duration is 14 days, there must be 2 pressurized rovers. If the crew duration is greater, there is no constraint on the type of pressurized mobility. (external requirement)
- Constraint #5: Outpost power must be present when there is habitation and if there is no habitation then no type of outpost power must be present. (incompatibility and sensibility)
- Constraint #6: If the crew duration is only 7 days, then there must be one (and only one) small unpressurized rover. (external requirement)
- Constraint #7: If the crew duration is only 7 days, there must be no enhanced communications (external requirement)
- Constraint #8: If the crew duration is only 7 days, there must be no ISRU systems (external requirement)

Each of these constraints limits the selection of alternatives for the connected decisions. As opposed to the visualization of the property functions as logical equations, shown in the previous chapter, Table 29 through Table 36 show an alternative visualization of the property functions known as logical tables. Each table has entries of 1s and 0s; a one signifies a feasible combination of the decisions and a zero signifies an infeasible combination.

Table 29. Logical table for Constraint 1 of the LSS model.

		mobilePower		
	Constraint #1	none	Solar array w energy storage	RPS
Press Mobility	none	1	0	0
	2x pressurized rover	0	1	1
	4x pressurized rovers	0	1	1

Table 30. Logical table for Constraint 2 of the LSS model.

		crewDuration				
	Constraint #2	7	14	28	60	180
habitation	none	1	1	0	0	0
	28d single module	0	0	1	0	0
	60d single module	0	0	0	1	0
	180d two module (assembled)	0	0	0	0	1
	180d two module (not assembled)	0	0	0	0	1

Table 31. Logical table for Constraint 3 of the LSS model.

		infrastructureMobility		
	Constraint #3	none	1x heavy lift mobility	2x heavy lift mobility
habitation	none	1	1	1
	28d single module	1	1	1
	60d single module	1	1	1
	180d two module (assembled)	0	1	1
	180d two module (not assembled)	1	1	1

Table 32. Logical table for Constraint 4 of the LSS model.

		pressMobility		
	Constraint #4	none	2x pressurized rover	4x pressurized rovers
crewDuration	7	1	0	0
	14	0	1	0
	28	1	1	1
	60	1	1	1
	180	1	1	1

Table 33. Logical table for Constraint 5 of the LSS model.

		outpostPower		
	Constraint #5	**none**	**Solar array w energy storage**	**Fission Surface Power**
habitation	**none**	1	0	0
	28d single module	0	1	1
	60d single module	0	1	1
	180d two module (assembled)	0	1	1
	180d two module (not assembled)	0	1	1

Table 34. Logical table for Constraint 6 of the LSS model.

		unpressMobility		
	Constraint #6	**none**	**1x small unpressurized**	**2x small unpressurized**
crewDuration	**7**	0	1	0
	14	1	1	1
	28	1	1	1
	60	1	1	1
	180	1	1	1

Table 35. Logical table for Constraint 7 of the LSS model.

		Communications	
	Constraint #7	**none**	**1x Communications Terminal**
crewDuration	**7**	1	0
	14	1	1
	28	1	1
	60	1	1
	180	1	1

Table 36. Logical table for Constraint 9 of the LSS model.

		ISRU	
	Constraint #8	**none**	**2x Oxygen Production Plants**
crewDuration	**7**	1	0
	14	1	1
	28	1	1
	60	1	1
	180	1	1

3.2.5 Summary of Representing

This section has outlined the formal steps in representing a system for single-level decision-based system architecture modeling. The LSS was represented using this framework to produce a system with 17 property variables, 11 decision variables, and 8 logical constraints. The complete representation of the LSS is shown in a tabular format in Table 37. The top of the tabular format shows how each of the decisions is connected to each other through the logical constraints, while the bottom rows show which decisions are connected to which property variables.

Table 37. Tabular representation of the LSS decision-based model.

		numCrewedMissions	crewDuration	numEVA	infrastructureMobility	pressurizedMobility	unpressurizedMobility	mobilityPower	habitation	outpostPower	communications	ISRU
		D1	D2	D3	D4	D5	D6	D7	D8	D9	D10	D11
numCrewedMissions	D1	■										
crewDuration	D2		■			C4	C6		C2		C7	C8
numEVA	D3			■								
infrastructureMobility	D4				■				C1			
pressurizedMobility	D5		C4			■		C3				
unpressurizedMobility	D6		C6				■					
mobilityPower	D7					C3		■				
habitation	D8		C2		C1				■	C5		
outpostPower	D9								C5	■		
communications	D10		C7								■	
ISRU	D11		C8									■
numAstronauts	M1	•										
num60dayMissions	M2	•	•									
totalCrewTime	M3	•	•	•								
EVACrewTime	M4	•	•	•								
IVACrewTime	M5	•	•	•								
explorationRange	M6				•	•	•					
pressurizedVolume	M7					•			•			
communications	M8										•	
ECLSSexp	M9	•	•						•			
roverOpsExp	M10	•	•			•						
ISRUexp	M11	•	•									•
powerExp	M12	•	•							•		
dustMitigationExp	M13	•	•	•								
numCargoFlights	M14	•	•	•	•	•	•	•	•	•	•	•
numCrewedFlights	M15	•										
utilizationMass	M16	•	•	•	•	•	•	•	•	•	•	•
lifeCycleCost	M17	•	•	•	•	•	•	•	•	•	•	•

3.3 Structural Reasoning

The second process of decision-based system architecture modeling is structural reasoning. This process is used to transform the information represented in the model into a sorted set of decisions by reasoning about the overall structure of the system. Table 38 shows the degree of connectivity, based on logical constraints, and the number of alternatives for each of the eleven decisions in the LSS model. From this table, it can be seen that the decisions of crew duration and the choice of habitation are the most highly connected to the other decisions (3 connections). From an architecting point of view, this information can be used to pick out which decisions are most influential on other decisions. In this example, making the crew duration and habitation decisions early will heavily influence the other decisions in the model, since they imply constraints on much of the rest of the architecture. The information related to the connectivity of the decisions can be used for engineering organization purposes. For example, the team considering habitation type must collaborate with all other teams considering the three other decisions to which habitation type is connected. This step provides partial information as to which decisions are important to make early. In section 3.5, we will see that the crew duration and habitation decisions are also important because it has a large effect on overall system benefit and cost.

Table 38. The degree of connectivity for each decision in the LSS model.

decision variable	degree of connectivity
crewDuration	5
habitation	3
pressurizedMobility	2
infrastructureMobility	1
unpressurizedMobility	1
mobilityPower	1
outpostPower	1
communications	1
ISRU	1
numCrewedMissions	0
numEVA	0

3.4 Simulating

The LSS decision-based system architecture model was compiled and executed using a value-based CSP solver developed in MatLab. The code used can be seen in the Appendix B.

Based on the chosen set of decisions and their respective alternatives, there were 218,700 unconstrained architectures. Through the application of the six constraints chosen, this created a set of 12,555 feasible architectures.

3.5 Viewing

The final process involved in decision-based system architecture modeling is viewing the information created by the model. The viewing process transforms the feasible combinations of decisions and their property variables into new available knowledge. In order for this information to be useful in the decision-making process, it must be presented in a way that is meaningful to the architect. The overall goal of viewing the simulation data is to improve the architect's ability to comprehend the space of feasible combinations of decisions. The following subsections describe the two main types of information that can be determined using a decision-based system architecture model:

- Identification of the preferred architecture
- Determination of the overall influence of a decision

3.5.1 Identifying the preferred architecture

Previous decision-based system architecture modeling methodologies suggested the use of Pareto Front Views to investigate which architectures were optimal [Sim08]. The limitation of Pareto Front Views is that they are ineffective for systems with a medium to large set of metrics. Because of this limitation, the use of Pareto Front Views was replaced with the application of Multi-Attribute Utility Theory (MAUT) in order to determine the 'best' architectures. This not only allowed an aggregate utility score to be calculated but also ensured that feasible architectures did not gain artificial advantages from high values in single metrics that provided no actual increase in benefit. Based on discussions with various experts, estimates were made for piecewise utility functions for each benefit as shown in Figure 11 through Figure 26.

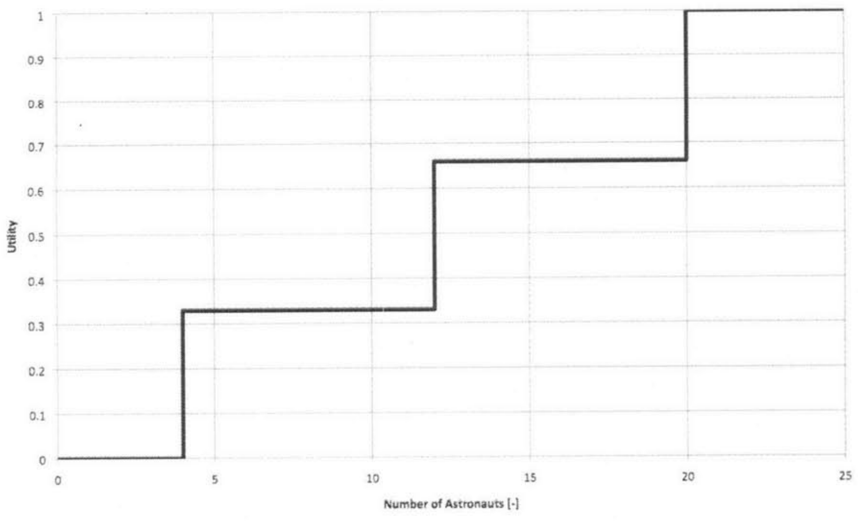


Figure 11. Utility Curve for Number of Astronauts.

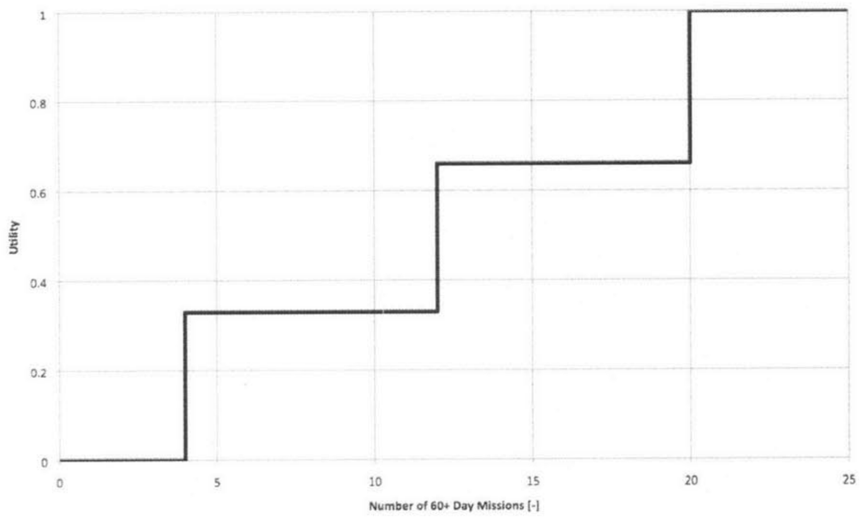


Figure 12. Utility Curve for Number of 60+ Day Missions.

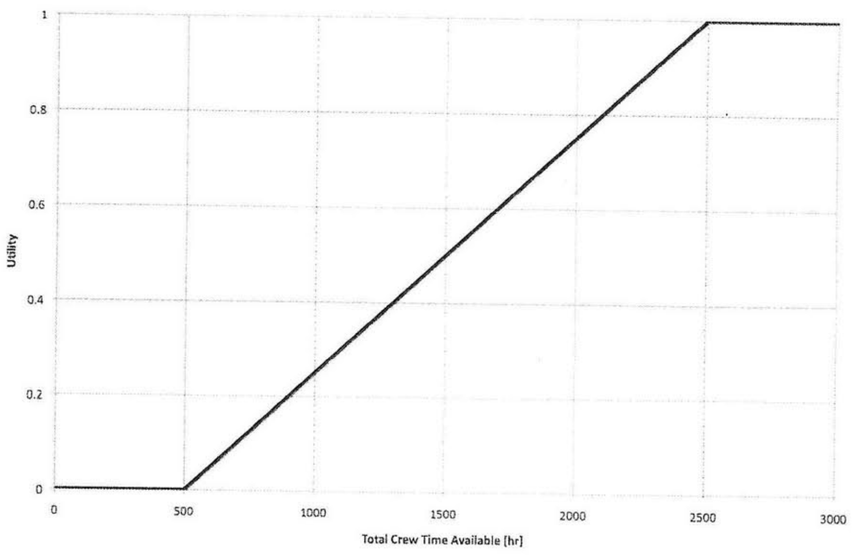


Figure 13. Utility Curve for Total Crew Time Available.

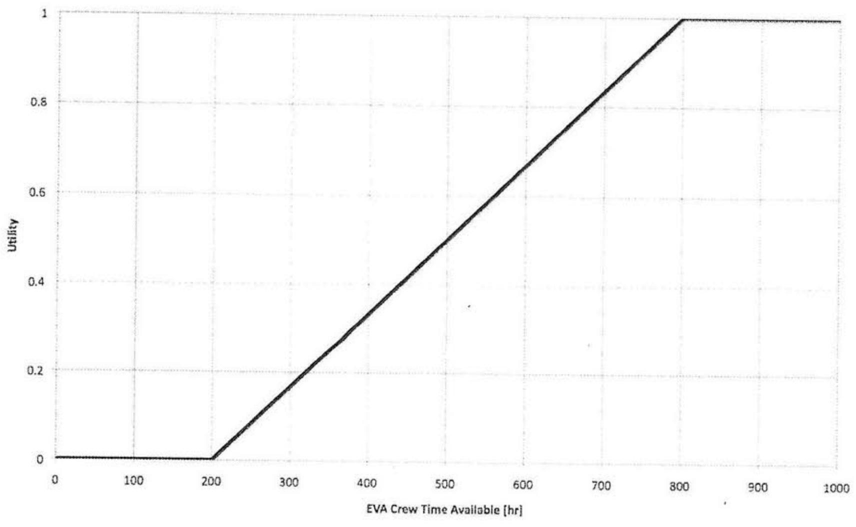


Figure 14. Utility Curve for EVA Crew Time Available.

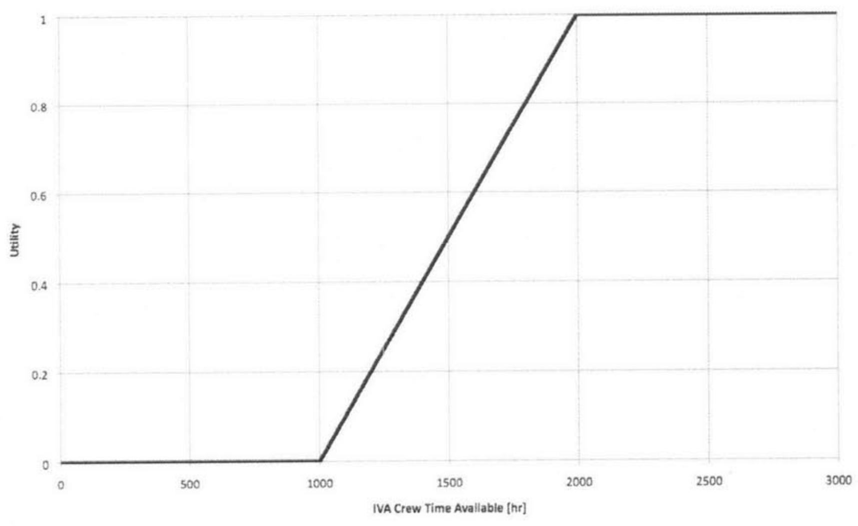


Figure 15. Utility Curve for IVA Crew Time Available.

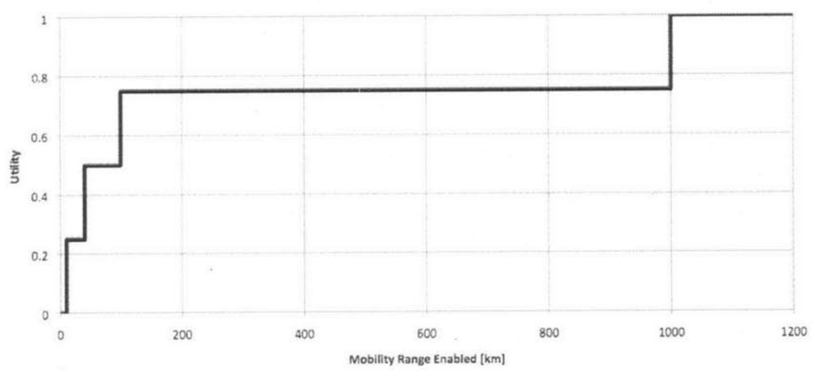


Figure 16. Utility Curve for Exploration Range Enabled.

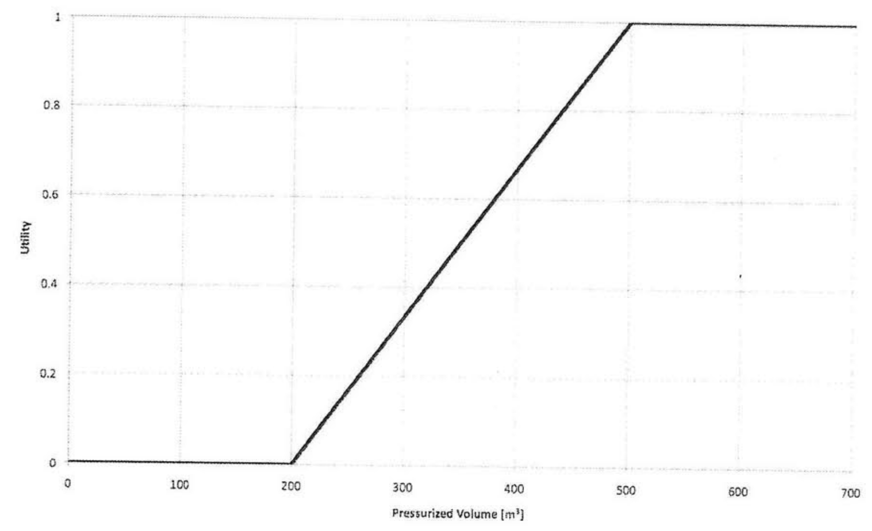


Figure 17. Utility Curve for Amount of Pressurized Volume.

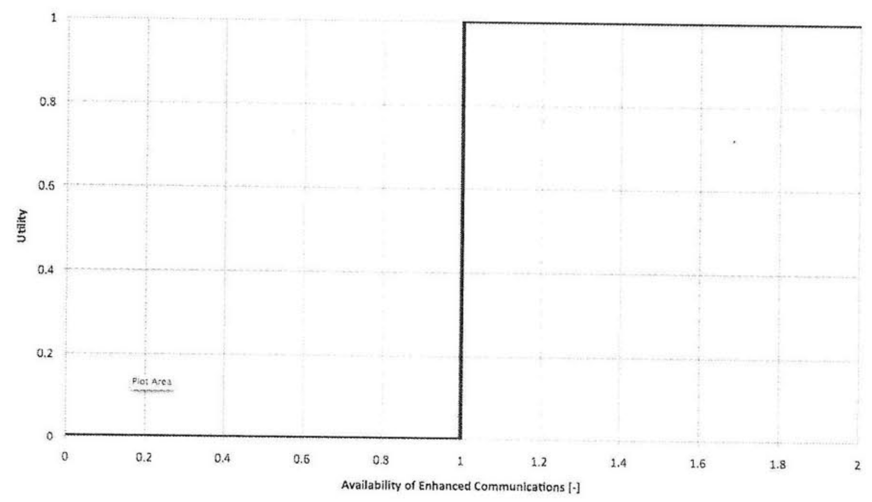


Figure 18. Utility Curve for Availability of Enhanced Communications.

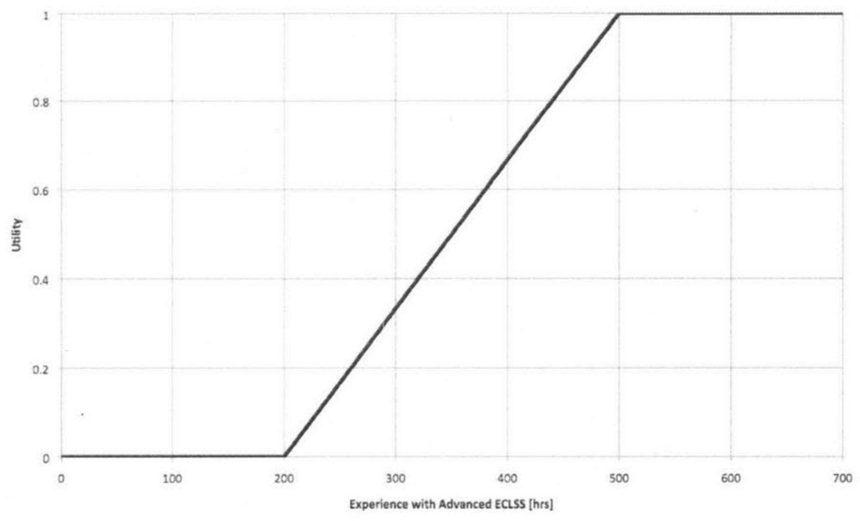


Figure 19. Utility Curve for Experience with Advanced ECLSS.

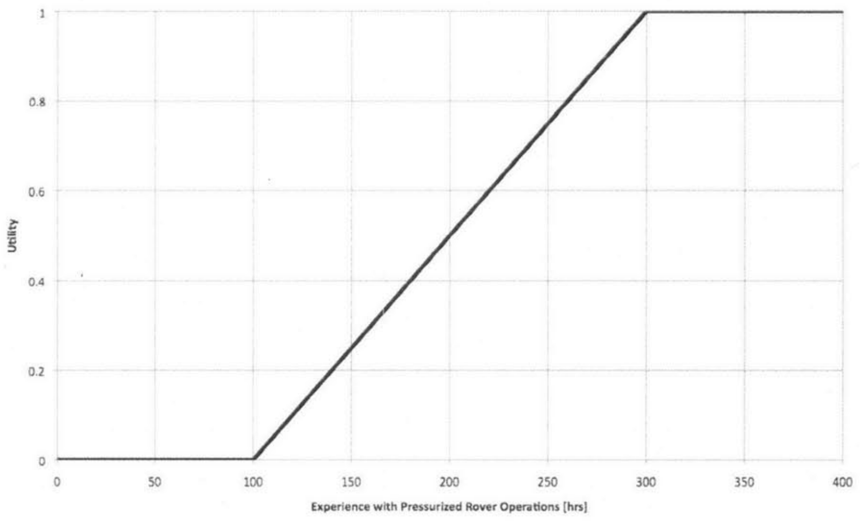


Figure 20. Utility Curve for Experience with Pressurized Rovers.

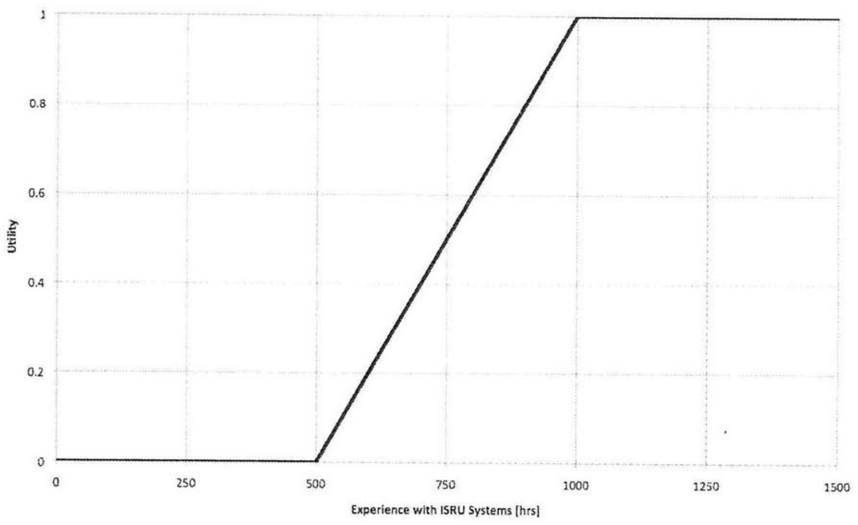


Figure 21. Utility Curve for Experience with ISRU Systems.

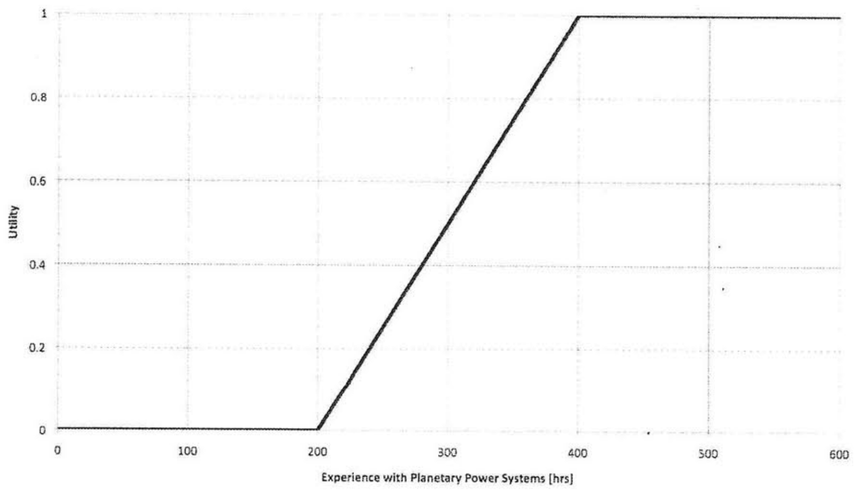


Figure 22. Utility Curve for Experience with Planetary Power Systems.

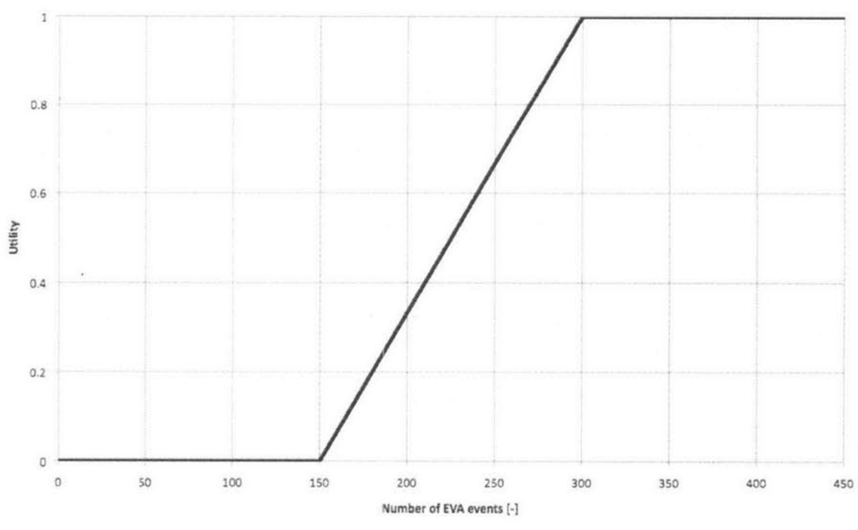


Figure 23. Utility Curve for Dust Mitigation Experience.

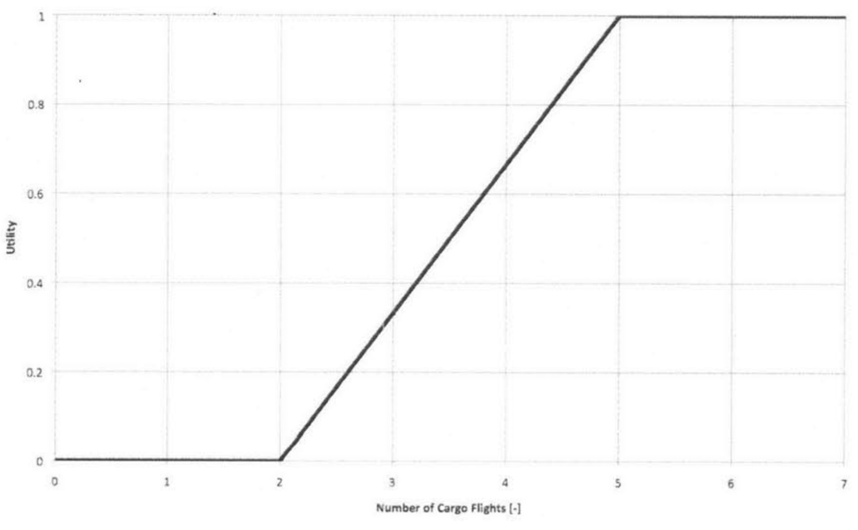


Figure 24. Utility Curve for Number of Cargo Flights.

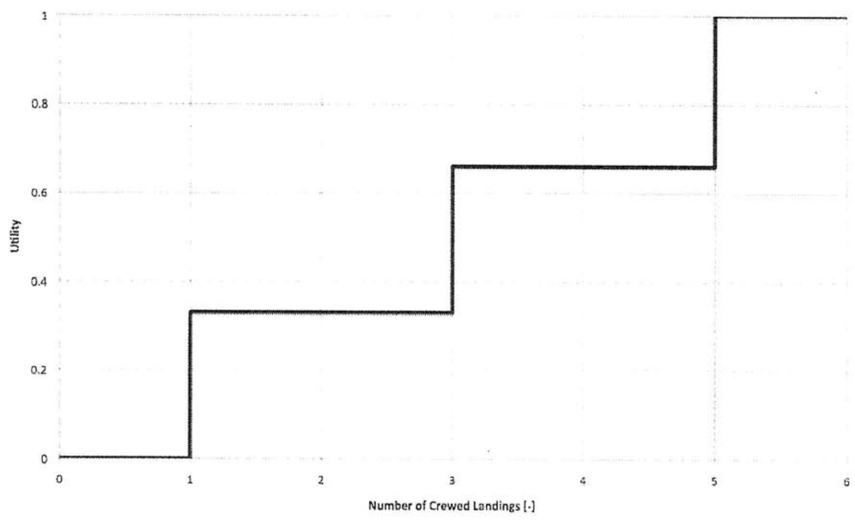


Figure 25. Utility Curve for Number of Crewed Landings.

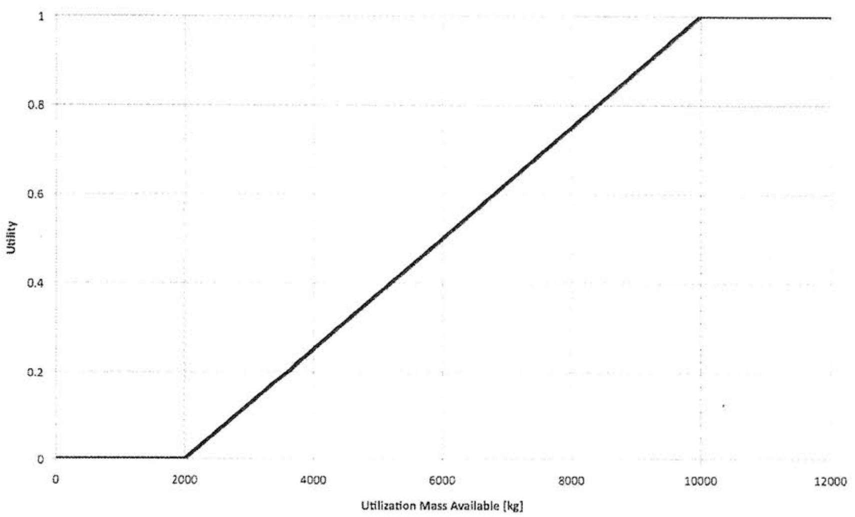


Figure 26. Utility Curve for Utilization Mass Available.

An aggregate utility score was calculated for the benefit of each architecture based on equal weighting of each benefit metric. This provides a reasonable starting point for the investigation of the architectures. Through discussions with stakeholders, the weightings of the utility scores can be altered to allow a better representation of the actual value of the system. Based on the application of this utility theory, 18 architectures were found to have a utility of 1.00. Of these 18 architectures, the one architecture with the lowest life cycle cost is shown in Table 39. This architecture also had a cost of \$36.09 billion USD (ranked as the 150 most expensive architecture out of 12,555).

Table 39. Description of LSS architecture with the highest utility.

Arch #	12227
Utility	1.00
Life-Cycle Cost	36.1 B\$ FY04
Number of Crewed Missions	5
Crew Duration	180
Number of EVA/wk	2
Infrastructure Mobility	2x heavy lift mobility
Pressurized Mobility	2x pressurized rover
Unpressurized Mobility	none
Mobility Power	Photo-voltaic with energy storage
Habitation	2-module assembled habitat
Outpost Power	Photo-voltaic with energy storage
Comm	1x LCT
ISRU	2x oxygen Production Plants

Figure 27 shows the Pareto Front View for life cycle cost versus architecture utility for all 12,555 feasible architectures. Table 40 through Table 42 show the top ranked architectures for the LSS in terms of highest benefit, lowest cost, and highest value (benefit/cost) respectively.

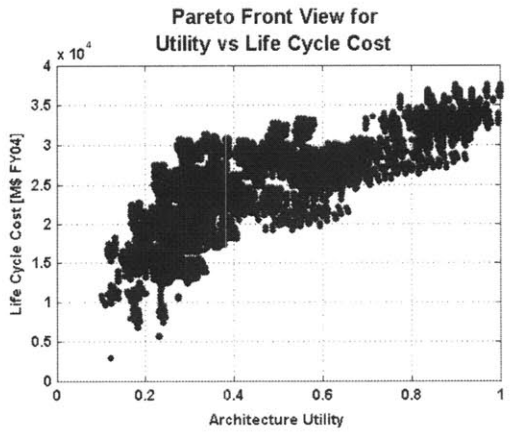


Figure 27. Pareto Front View for Life Cycle Cost vs. Utility for the LSS.

Table 40. Top 18 architectures based on benefit score for the LSS.

numMissions	duration [days]	numEVAs/wk	infraMobility	pressMobility	unpressMobility	mobilePower	habitation	outpostPower	comm	ISRU	Cost score	Benefit score	Value score
5	180	3	1x	4x	none	PV with storage	180day dual (assem)	fission system	1x	2x	0.96	1.00	1.04
5	180	3	1x	4x	none	PV with storage	180day dual (unassem)	fission system	1x	2x	0.96	1.00	1.04
5	180	3	1x	4x	1x	PV with storage	180day dual (assem)	fission system	1x	2x	0.97	1.00	1.03
5	180	3	1x	4x	1x	PV with storage	180day dual (unassem)	fission system	1x	2x	0.97	1.00	1.03
5	180	3	1x	4x	2x	PV with storage	180day dual (assem)	fission system	1x	2x	0.98	1.00	1.02
5	180	3	1x	4x	2x	PV with storage	180day dual (unassem)	fission system	1x	2x	0.98	1.00	1.02
5	180	3	1x	4x	none	RTG	180day dual (assem)	fission system	1x	2x	0.98	1.00	1.02
5	180	3	1x	4x	none	RTG	180day dual (unassem)	fission system	1x	2x	0.98	1.00	1.02
5	180	3	2x	4x	none	RTG	180day dual (assem)	fission system	1x	2x	0.98	1.00	1.02
5	180	3	2x	4x	none	RTG	180day dual (unassem)	fission system	1x	2x	0.98	1.00	1.02
5	180	3	1x	4x	1x	RTG	180day dual (assem)	fission system	1x	2x	0.99	1.00	1.01
5	180	3	1x	4x	1x	RTG	180day dual (unassem)	fission system	1x	2x	0.99	1.00	1.01
5	180	3	2x	4x	1x	RTG	180day dual (assem)	fission system	1x	2x	0.99	1.00	1.01
5	180	3	2x	4x	1x	RTG	180day dual (unassem)	fission system	1x	2x	0.99	1.00	1.01
5	180	3	1x	4x	2x	RTG	180day dual (assem)	fission system	1x	2x	1.00	1.00	1.00
5	180	3	1x	4x	2x	RTG	180day dual (unassem)	fission system	1x	2x	1.00	1.00	1.00
5	180	3	2x	4x	2x	RTG	180day dual (assem)	fission system	1x	2x	1.00	1.00	1.00
5	180	3	2x	4x	2x	RTG	180day dual (unassem)	fission system	1x	2x	1.00	1.00	1.00

Table 41. Top 10 architectures based on lowest cost scores for the LSS.

numMissions	duration [days]	numEVAs/wk	infraMobility	pressMobility	unpressMobility	mobilePower	habitation	outpostPower	comm	ISRU	Cost score	Benefit score	Value score
1	7	1	none	none	1x	none	none	none	none	none	0.08	0.12	1.55
1	7	2	none	none	1x	none	none	none	none	none	0.08	0.12	1.55
1	7	3	none	none	1x	none	none	none	none	none	0.08	0.12	1.55
1	14	1	none	2x	none	PV with storage	none	none	none	none	0.18	0.18	1.00
1	14	2	none	2x	none	PV with storage	none	none	none	none	0.18	0.18	1.00
1	14	3	none	2x	none	PV with storage	none	none	none	none	0.18	0.18	1.00
1	14	1	none	2x	1x	PV with storage	none	none	none	none	0.19	0.18	0.93
1	14	2	none	2x	1x	PV with storage	none	none	none	none	0.19	0.18	0.93
1	14	3	none	2x	1x	PV with storage	none	none	none	none	0.19	0.18	0.93
1	14	1	none	2x	none	PV with storage	none	none	1x	none	0.20	0.24	1.21

Table 42. Top 10 LSS architectures based on value score.

numMissions	duration [days]	numEVAs/wk	infraMobility	pressMobility	unpressMobility	mobilePower	habitation	outpostPower	comm	ISRU	Cost score	Benefit score	Value score
1	7	3	-	-	1x	-	-	-	-	-	0.08	0.12	1.55
1	7	2	-	-	1x	-	-	-	-	-	0.08	0.12	1.55
1	7	1	-	-	1x	-	-	-	-	-	0.08	0.12	1.55
5	180	3	1x	-	-	-	180day dual (assem)	PV with storage	1x	2x	0.75	0.92	1.23
5	180	3	1x	-	-	-	180day dual (unassem)	PV with storage	1x	2x	0.75	0.92	1.23
1	14	3	-	2x	-	PV with storage	-	-	1x	-	0.20	0.24	1.22
1	14	2	-	2x	-	PV with storage	-	-	1x	-	0.20	0.24	1.22
1	14	1	-	2x	-	PV with storage	-	-	1x	-	0.20	0.24	1.21

3.5.2 Determining the overall influence of a decision

A benefit, cost, and value Average Influence Scores can be calculated for each decision in the LSS model using the set of architectures on the Pareto Front. Based on the 16 benefit property variables and the single cost property variable chosen, the most influential decisions are the duration of crewed missions and the choice of habitation and the least influential decisions are the number of missions and the number of EVAs as shown in Table 43.

Table 43. The Average Influence Scores for the eleven decisions of the LSS model.

Decision Variable	Benefit AIS	Cost AIS	Value AIS
duration	0.80	0.50	0.65
habitat	0.48	0.63	0.56
pressMobility	0.34	0.64	0.49
infraMobility	0.23	0.60	0.41
comm	0.23	0.55	0.39
ISRU	0.23	0.56	0.39
unpressMobility	0.34	0.44	0.39
mobilePower	0.24	0.53	0.38
outpostPower	0.31	0.38	0.35
numMission	0.20	0.40	0.30
numEVA	0.13	0.45	0.29

3.6 Summary

This chapter has described the application of single-level decision-based system architecture modeling to an investigation of the Lunar Surface System (LSS). The LSS is an interesting case study to investigate since it is a complex system with both a large number of property variables and a large number of interconnected functions. Section 3.2 described the formulation of the LSS trade space as a decision-based model and showed how the system could be represented with 17 property variables, 11 decisions, and 8 logical constraints. Section 3.3 and Section 3.4 discussed the structural reasoning and simulation steps associated with developing a decision-based model. Section 3.5 showed the results of the model in terms of the interesting architectures as well as the most influential decisions.

4 Deep Space Design Reference Missions: A Case Study in Single-level Decision-Based System Architecture Modeling

4.1 Introduction

Following the findings of the Review of U.S. Human Spaceflight Plans Committee [Aug09], the FY2011 budget proposal for NASA set a new direction for NASA's human spaceflight program [NASA10]. As part of this shift in direction, NASA's Exploration Systems Mission Directorate (ESMD) was charged with developing a new set of design reference missions (DRMs) for human exploration in order to set the priorities for the program based upon the needs and requirements of these DRMs. As part of this "innovative new path", the budget proposal directed NASA to investigate multiple potential destinations "including the Moon, asteroids, Lagrange points, and Mars and its environs". Based on these DRMs and the research and technology investments they recommend, NASA will set the many near-term steps required to eventually enable missions that send humans beyond LEO. As part of this effort, NASA would benefit from being able to comprehensively investigate the related trade spaces for each DRM and gain insight in which architecture decisions are the most influential.

The objective of this chapter is to describe the case study of applying single-level decision-based system architecture modeling to investigating the trade space for deep space DRMs in order to identify both the interesting system architectures and the influential Level 1 decisions.

As discussed in Chapter One and mentioned in the FY 2011 NASA budget proposal, there are several destinations beyond the Earth-Moon neighborhood (i.e. in deep space) that humans could visit in the inner solar system. These destinations include Sun-Earth Lagrange Points, Near Earth Objects, and the vicinity of Mars including Phobos and Deimos. The case study in this chapter focuses on Near Earth Objects as the destination of interest, however, the mission profile used for NEO missions, shown in Figure 28, can be used for any of the deep space destinations.

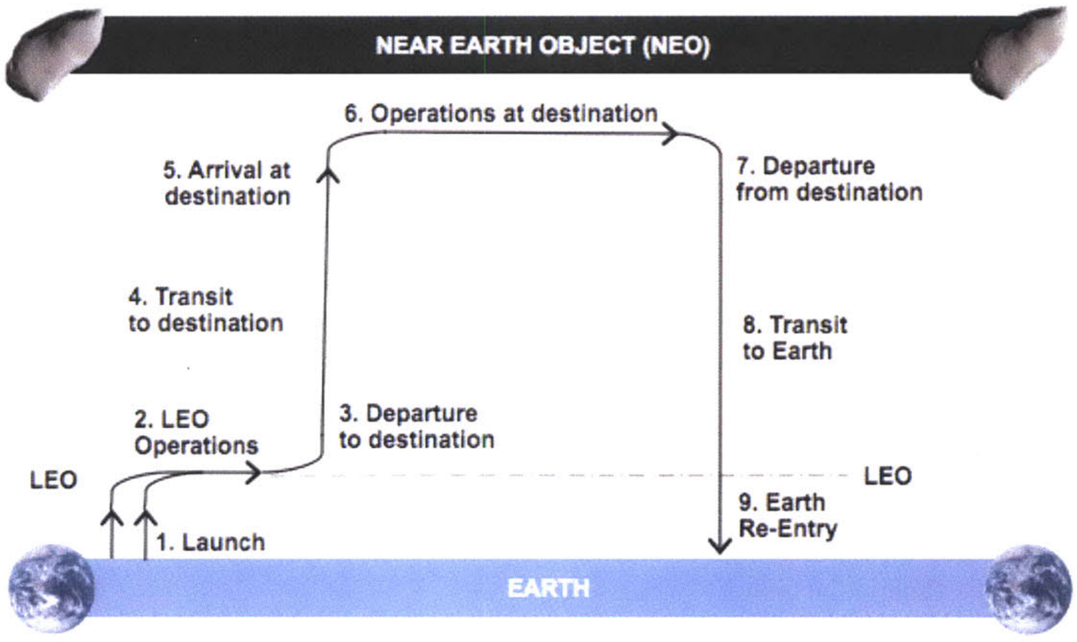

NEAR EARTH OBJECT (NEO)
6. Operations at destination
5. Arrival at destination
7. Departure from destination
4. Transit to destination
8. Transit to Earth
2. LEO Operations
3. Departure to destination
LEO
LEO
9. Earth Re-Entry
1. Launch
EARTH

Figure 28. Generic mission profile for all NEO missions.

The nine operations shown in the figure are present in all missions to deep space destinations. The nine operations are as follows:

1. Launch to LEO: All crew and cargo are launched from the surface of the Earth to LEO. This can be completed in a single or a combination of launches depending on the requirements and the launch strategy.
2. LEO Operations: Depending on the launch strategy, several operations may need to occur in LEO before the mission can continue. These operations include assembly of elements and refueling or transfer of propellant as required.
3. Departure to destination: A propulsive maneuver required to provide energy to the assembled spacecraft to send the mission on a trajectory to its given destination. A characteristic of the maneuver is the amount of delta-v that is required.

4. Transit to Destination: In-space operations that occur between the time the spacecraft leave LEO until it arrives into the vicinity of the destination. A characteristic of this operation is the duration (in days) of the transit.
5. Arrival at Destination: A propulsive maneuver required to provide energy to the assembled spacecraft to allow the mission to remain within the vicinity of the destination (either in an orbit or in a parallel trajectory). A characteristic of the maneuver is the amount of delta-v that is required.
6. Operations at Destination: In-space operations that occur while the spacecraft is in the vicinity of the destination. These operations can include scientific exploration, public engagement activities, or testing of exploration systems and operations. A characteristic of this operation is the duration (in days) of the time in the vicinity of the destination.
7. Departure from Destination: A propulsive maneuver required to provide energy to the assembled spacecraft to send the mission on a return trajectory towards Earth. A characteristic of the maneuver is the amount of delta-v that is required.
8. Transit to Earth: In-space operations that occur between the time the spacecraft leave the vicinity of the destination until it begins the re-entry procedure into Earth's atmosphere. A characteristic of this operation is the duration (in days) of the time required for the transit.
9. Re-entry at Earth: The return of the astronauts through the Earth's atmosphere for a safe landing on land or at sea. The re-entry is assumed to be performed by a suitably sized crew capsule capable of supporting the necessary number of crew.

This case study focuses on the operations that occur once all the assets are launched and assembled (i.e. operation 3 onwards). For these operations, regardless of the destination, it is only the characteristics, such as durations and delta-v, which changes from mission to mission. The characteristics for four representative proposed NEO missions are shown in Table 44.

Table 44. Mission characteristics for various proposed NEO missions.

Mission type	3. Earth Departure (delta v in km/s)	4. Transit to destination (duration in days)	5. Destination Arrival (delta v in km/s)	6. Destination operations (duration in days)	7. Destination departure (delta v in km/s)	8. Transit to Earth (duration in days)
NEO 1991 VG (sprint) [in 2017]	3.75	29	5.01	30	4.05	30
NEO 1991 VG [in 2007]	3.37	70	1.21	16	1.16	89
NEO 2001 GP2 [in 2019]	1.54	130	2.09	14	0.17	160
NEO 1999 AO10 [in 2025]	3.29	111	2.19	14	1.75	31

The specific objectives for this chapter focus on the steps required to develop a single-level decision-based system architecture model to explore the trade space for the in-space operations (operations 3 to 9 on Figure 28) for the four possible human NEO missions with the characteristics given in Table 44. These objectives include:

- Representing the system as a set of interconnected decisions with appropriate property variables, property functions, and logical constraints
- Reasoning about the structure of the system to provide insight to assist in the simulation of the model
- Simulating the system as represented in order to enumerate and evaluate all feasible architectures for the given NEO missions
- Viewing the data created by the model in order to identify interesting architectures for the missions and gain insight into which decisions are the most influential at the system-level.

The next four sections of this chapter outline the four steps in developing a single-level decision-based system architecture model as described in chapter Two. Section 4.2

outlines the four tasks required in representing the system architectures feasible for a human NEO mission. Section 4.3 discusses at the structure of the system and provides insight based on the structure. Section 4.4 details how the decision-based model was simulated. Section 4.5 details the information gained from the model, both in terms of identifying interesting architectures as well as gaining insight into the influence of the architectural decisions.

4.2 Representing

This section describes the details in each of the four steps required to represent the design space for a human NEO mission as a decision-based model. These steps include:

- Choosing the property variables.
- Choosing the decision variables.
- Enumerating the property functions.
- Enumerating the logical constraints.

Each of the following subsections focuses on a single task in the process.

4.2.1 Choosing the property variables

In order to evaluate any of the feasible architectures enumerated by the decision-based model, a set of property variables must be selected to capture both the benefit and the cost sides of the value equation for the system. Using the reasoning outlined in section 2.2.1 regarding the decomposition of the needs and goals of the system to define a set of qualitative property variables, seven property variables were chosen. Four of these property variables provide a first order approximation for the benefit taken from the system, as shown in Table 45, and three of the property variables capture a first order approximation for the potential cost of performing a human NEO mission, as shown in Table 46.

Table 45. Benefit property variables for the NEO mission-level model.

Property Variable	Description	Reasoning
Number of Crew-Days at the NEO	The total number of crew-days available while within the vicinity of the NEO.	The more crew-days at the NEO, the more scientific exploration can occur.
Number of Crew-Days in Space	The total number of crew-days available throughout the entire mission.	The more crew-days available in space, the more scientific experimentation and public engagement can occur.
Useful Payload Mass Available [mt]	The total mass available for useful payload.	The more useful payload available, the more opportunities for scientific equipment as well as third party payloads (e.g. educational, commercial, international).
Methane Propulsion Development	Binary variable (1=yes, 0=no) describing whether or not methane propulsion has been developed.	The development of methane propulsion is positive as it benefits the feasibility of propellant refueling to support future exploration.

Table 46. Cost property variables for the NEO mission-level model.

Property Variable	Description	Reasoning
Initial Mass in Low Earth Orbit (IMLEO) [mt]	The mass of the in space system required to support the mission.	The larger the system mass, the higher the related launch costs.
Number of Unique Project Required	The total number of unique development projects that must occur for the mission to be feasible.	The more unique projects the required, the higher the development costs for the mission.
Development of Nuclear Propulsion	Binary variable (1=yes, 0=no) describing whether or not nuclear propulsion has been developed.	The development of nuclear propulsion increases the cost of the system as nuclear propulsion is seen as a substantial undertaking if required.

4.2.2 Choosing the decision variables

The second step in representing the system as a decision-based model is selecting the decision variables and their feasible alternatives. As mentioned in previous chapters, there are two sets of decisions that must be developed:

- A set of decisions related to selecting the attribute for the system.
- A set of decisions related to mapping forms to functions.

The model includes three system attribute decisions to bound the trade space in terms of the overall concept of the mission. These decisions include:

- Which mission opportunity is targeted? This decision selects which of the four representative NEO missions is being investigated.
- What is the number of crew on the mission? This decision varies how many crew are available during the mission.
- How much useful payload is budgeted? This decision investigates different options for the mass of useful payload that is carried to the NEO in order to support activities such as scientific exploration and public engagement.

In addition to these three system parameter decisions, the model includes another three system attribute decisions included in order to allow an investigation of an interesting aspect of the trade space: which propulsion types are developed and what type of commonality is implemented between the propulsion elements. These three system attribute decisions include:

- What type of commonality is implemented for the propulsion elements? This decision explores the impact of enforce none, limited, or full commonality in terms of the class of propulsion used for the three major propulsive maneuvers.
- Is nuclear propulsion developed? This decision investigates the possibility that nuclear propulsion may or may not be developed to support the NEO missions.
- Is methane propulsion developed? This decision investigates the possibility that methane propulsion may or may not be developed to support the NEO missions.

The six system attribute decision variables and their chosen alternatives are shown in Table 47.

Table 47. System parameter decision variables and alternatives for the NEO mission-level model.

Decision	Alt A	Alt B	Alt C	Alt D	Alt E	Alt F
Which mission opportunity is targeted?	156 day mission in 2025	304 day mission in 2019	89 day mission in 2017	175 day mission in 2007		
What is the number of crew?	2	4	6			
How much useful payload is budgeted?	2	4	6	8	10	
What type of commonality is implemented among the propulsion systems?	No commonality	Partial (2 out of 3) commonality	Full commonality			
Is nuclear propulsion developed?	yes	no				
Is methane propulsion developed?	yes	no				

In terms of the form to function mapping decision variables, the model includes five specific decisions related to the following five internal functions required for the mission:

- Providing propulsion capability to depart LEO.
- Providing habitation capability to support the crews while in-space.
- Providing propulsion capability to arrive at the NEO.
- Providing propulsion capability to depart from the NEO.
- Providing re-entry capability for the crew upon their return to Earth.

Table 48 shows the five form-function mapping decisions included in the model and their respective alternatives. The habitation and re-entry related decisions are populated with specific elements as was done in the LSS case study. The three propulsion related decisions have alternatives that set the type of propulsion system and allows the actual size of the elements to be sized using the characteristics of the respective alternatives and the overall mass of the system as shown in the next subsection.

Table 48. Form-function mapping decision variables and alternatives for the NEO mission-level model.

Decision	Alt A	Alt B	Alt C	Alt D	Alt E	Alt F
How is propulsion for departing from LEO provided?	LOX/LH2 stage	LOX/LCH4 stage	NTR stage			
How is in-space habitation provided?	2crew 90day habitat	4crew 90day habitat	6crew 90day habitat	2crew 180day habitat	4crew 180day habitat	6crew 180day habitat
How is propulsion for arriving at the NEO provided?	LOX/LH2 stage	LOX/LCH4 stage	NTR stage			
How is propulsion for departing from the NEO provided?	LOX/LH2 stage	LOX/LCH4 stage	NTR stage			
How is the capability for Earth re-entry provided?	2person vehicle	4person vehicle	6person vehicle			

4.2.3 Enumerating the property functions

The third step in representing a system architecture problem as a decision-based model is to enumerate the property functions. Since this model has seven property variables, seven property functions must be enumerated. Table 49 shows the equations used for the seven property functions and Table 50 and Table 51 show the related characteristics required for their calculation as discussed in Chapter Two.

Table 49. Property functions for the NEO mission-level model.

Property Variable	Property Functions
Number of crew-days at the NEO	Number of crew characteristic x duration at NEO characteristic
Number of crew-days in space	Number of crew characteristic x duration in space characteristic
Useful payload available [mt]	Useful payload available characteristic
Development of methane propulsion	Development of methane propulsion characteristic
Initial Mass in Low Earth Orbit (IMLEO) [mt]	habitat mass characteristic + re-entry mass characteristic + useful payload available characteristic + destination departure propulsion mass + destination arrival propulsion mass + earth departure propulsion mass
# of unique projects required	sum(project development characteristic)
Development of nuclear propulsion	Development of nuclear propulsion characteristic

The only property function that is challenging to comprehend is related to the property variable of IMLEO. The property function is the sum of the masses of each element included in the given architecture. For the habitat, re-entry vehicle, and useful payload, these masses are simply characteristics of the chosen alternatives for the architecture, however; as mentioned above, the three propulsion elements are sized based on the overall mass of the system. Assuming the habitat, re-entry vehicle, and useful payload are carried through all three propulsive burns, each the mass for each propulsion element is calculated using the relevant characteristics, of specific impulse (in seconds) and inert mass fraction for the chosen alternative, in conjunction with the rocket equation [Cho96]. Each propulsion element is sized (both its propellant required and its inert mass) based on Equation 3.

Equation 3. Rocket equation calculation using delta-v, specific impulse, and inert mass fraction.

$$m_f = m_i \times e^{\frac{-\Delta V}{9.81 \times I_{sp}}}$$

$$where$$

$$m_f : mass\,final(kg),\ m_i : mass\,initial(kg)$$

$$\Delta V : Delta - V\,required(m/s),\ I_{sp} : \text{Specific Impulse(sec)}$$

$$m_f = m_{pl} + \alpha \times m_{prop}$$

$$m_i = m_{pl} + m_{prop} + \alpha \times m_{prop}$$

$$m_{pl} : \text{payload mass (kg)},\ m_{prop} : \text{propellant mass (kg)}$$

$$\alpha : \text{Inert Mass Fraction (inert mass/propellant mass)}$$

Table 50. Characteristics for the NEO mission-level model (1 of 2).

		Characteristics								
Decision Variables	**Decision Alternatives**	**Duration at NEO [days]**	**Duration in space [days]**	**Number of crew [-]**	**Useful payload available [mt]**	**Methane Prop Development**	**Specific Impulse [sec]**	**Inert Mass Fraction [-]**	**Element Mass**	**Nuclear Prop Development**
Which mission opportunity is targeted?	156 day mission in 2025	14	156							
	304 day mission in 2019	14	304							
	89 day mission	30	89							
	175 day mission	16	175							
What is the number of crew?	2			2						
	4			4						
	6			6						
What type of commonality is implemented among the propulsion systems?	none									
	in-space only									
	full									
How much useful payload is budgeted?	2				2					
	4				4					
	6				6					
	8				8					
	10				10					
Is nuclear propulsion developed?	yes									1
	no									0
Is methane propulsion developed?	yes					1				
	no					0				

Table 51. Characteristics for the NEO mission-level model (2 of 2).

		Characteristics								
Decision Variables	**Decision Alternatives**	**Duration at NEO [days]**	**Duration in space [days]**	**Number of crew [-]**	**Useful payload available [mt]**	**Methane Prop Development**	**Specific Impulse [sec]**	**Inert Mass Fraction [-]**	**Element Mass**	**Nuclear Prop Development**
How is propulsion for departing from LEO provided?	LOX/LH2 stage						450	0.15		
	LOX/LCH4 stage						369	0.15		
	NTR stage						900	0.78		
How is in-space habitation provided?	2person 90day habitat								18	
	4person 90 day habitat								25	
	6person 90day habitat								33	
	2person 180day habitat								21	
	4person 180day habitat								32	
	6person 180day habitat								43	
How is propulsion for arriving at the NEO provided?	LOX/LH2 stage						450	0.15		
	LOX/LCH4 stage						369	0.15		
	NTR stage						900	0.78		
How is propulsion for departing from the NEO provided?	LOX/LH2 stage						450	0.15		
	LOX/LCH4 stage						369	0.15		
	NTR stage						900	0.78		
How is the capability for Earth re-entry provided?	2person vehicle								12	
	4person vehicle								14	
	6person vehicle								16	

4.2.4 Enumerating the logical constraints

The final task in representing the system as a decision-based model is to enumerate the logical constraints that ensure that only feasible combinations of decisions (i.e. feasible architectures) are enumerated and evaluated during the simulation. This is completed by the inclusion of the relevant logical constraints. The NEO Level 1 model includes six logical constraints:

- Constraint 1: The habitat MUST be capable of supporting at least the desired number of crew and the habitat WILL NOT be oversized for the number of crew (logical incompatibility and sensibility).
- Constraint 2: The habitat MUST be capable of supporting the duration of the mission and the habitat WILL NOT be oversized in terms of the duration it is capable of supporting (logical incompatibility and sensibility). [Note: based on the literature, it is assumed any habitat capable of supporting a crew for 180 days can also support them for any higher number of days as well.]
- Constraint 3: Two of the three propulsion systems must be similar if partial propulsion commonality is enforced and all three propulsion systems must be of the same type if full propulsion commonality is enforced and no propulsion elements can be similar if there is no propulsion commonality (logical incompatibility)
- Constraint 4: The re-entry vehicle MUST be able to support, at minimum, the number of crew for the mission (logical incompatibility).
- Constraint 5: If nuclear propulsion is developed, at least one propulsion system MUST be nuclear and if nuclear propulsion is not developed, all propulsion systems MUST NOT be nuclear.
- Constraint 6: If methane propulsion is developed, at least one propulsion system MUST be methane and if methane propulsion is not developed, all propulsion systems MUST NOT be methane.

Table 52 shows the equations for each logical constraint in the system. With the property variables, decision variables, property functions, and logical constraints selected, the Level 1 decision-based model for the system architecture of NEO missions is complete. The next step in the process is to reason about the structure of the system.

Table 52. Logical constraints for the NEO mission-level model.

Constraint Name	Decision Variables Impacted	Logical Constraint Equations
Constraint 1	numCrew, hab	(numCrew==A&&(hab==A||hab==D))|| (numCrew==B&&(hab==B||hab==E))|| (numCrew==C&&(hab==C||hab==F))
Constraint 2	missionType,hab	missionType==A||missionType==B|| (missionType==D&&(hab==D||hab==E||hab==F))|| (missionType==C&&(hab==A||hab==B||hab==C))
Constraint 3	propCommonality, destDept,destArr, earthDept	(propCommonality==A&&(earthDept~=destArr)&& (earthDept~=destDept)&&(destArr~=destDept))|| (propCommonality==B&&((earthDept==destArr&& earthDept~=destDept)||(earthDept==destDept&& earthDept~=destArr)||(destDept==destArr&& earthDept~=destArr))|| (propCommonality==C&&destDept==earthDept&& destDept==destArr)
Constraint 4	numCrew,reEntry	(numCrew==A)||(numCrew==B&&reEntry~=A)|| (numCrew==C&&reEntry==C)
Constraint 5	nucPropDev, earthDept,destArr, destDept	(nucPropDev==A&& (earthDept==C||destArr==C||destDept==C))|| (nucPropDev==B&& (earthDept~=C&&destArr~=C&&destDept~=C))
Constraint 6	methPropDev, earthDept,destArr, destDept	(methPropDev==A&& (earthDept==B||destArr==B||destDept==B))|| (methPropDev==B&& (earthDept~=B&&destArr~=B&&destDept~=B))

4.3 Structural Reasoning

The second process of decision-based system architecture modeling is structural reasoning. This process is used to transform the information represented in the model into a sorted set of decisions by reasoning about the overall structure of the system. Table 53 shows the degree of connectivity, based on logical constraints, and the number of alternatives for each of the eleven decisions in the NEO Level 1 model. From this table, it can be seen that the decisions related to the choice of propulsion type for the three propulsive elements are the most highly connected to the other decisions (3 connections).

Table 53. Number of connections for each decision in the NEO mission model.

Decision Variables	Number of connections	Number of Alternatives
missionType	1	4
numCrew	2	3
propCommonality	1	3
usefulPayload	0	5
nucPropDev	1	2
methPropDev	1	2
earthDept	3	3
hab	2	6
destArr	3	3
destDept	3	3
reEntry	1	2

4.4 Simulating

The NEO mission-level decision-based system architecture model was compiled and executed using a value-based CSP solver developed in MatLab. The code used can be seen in the Appendix C.

Based on the chosen set of decisions and their respective alternatives, there were 349,920 unconstrained architectures. Through the application of the six constraints chosen, this created a set of 3,240 feasible architectures.

4.5 Viewing

The final process involved in decision-based system architecture modeling is viewing the information created by the model. The viewing process transforms the feasible combinations of decisions and their property variables into new available knowledge. In order for this information to be useful in the decision-making process, it must be presented in a way that is meaningful to the architect. The overall goal of viewing the simulation data is to improve the architect's ability to comprehend the space of feasible combinations of decisions. The following subsections describe the two main types of information that can be determined using a decision-based system architecture model:

- Identification of the preferred architectures
- Determination of the overall influence of a decision

4.5.1 Identifying the preferred architectures

In order to identify the preferred system architectures for the design of a human NEO mission, utility curves were developed for each of the seven property variables. Figure 29 through Figure 35 show the seven utility curves.

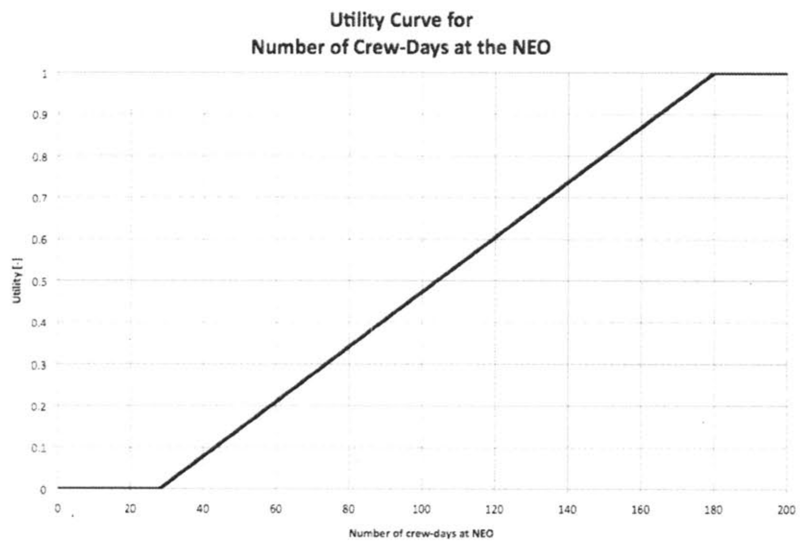


Figure 29. Utility curve for the number of crew-days at the NEO.

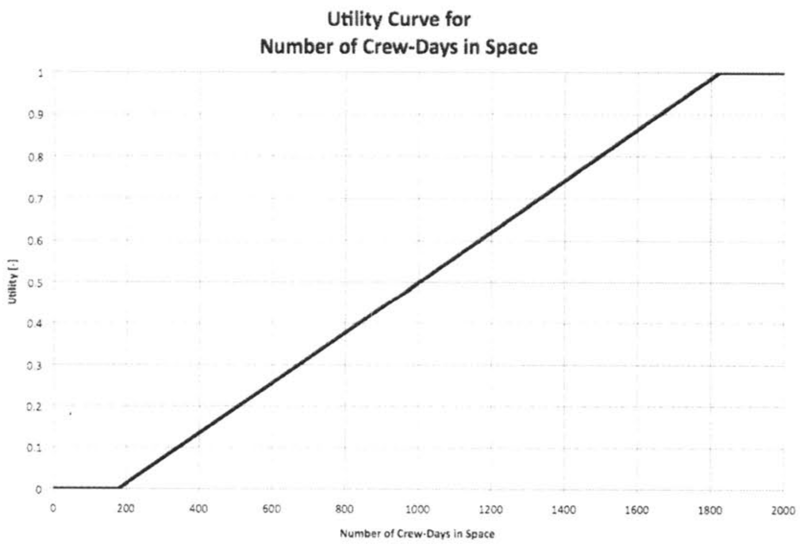


Figure 30. Utility curve for the number of crew-days in space.

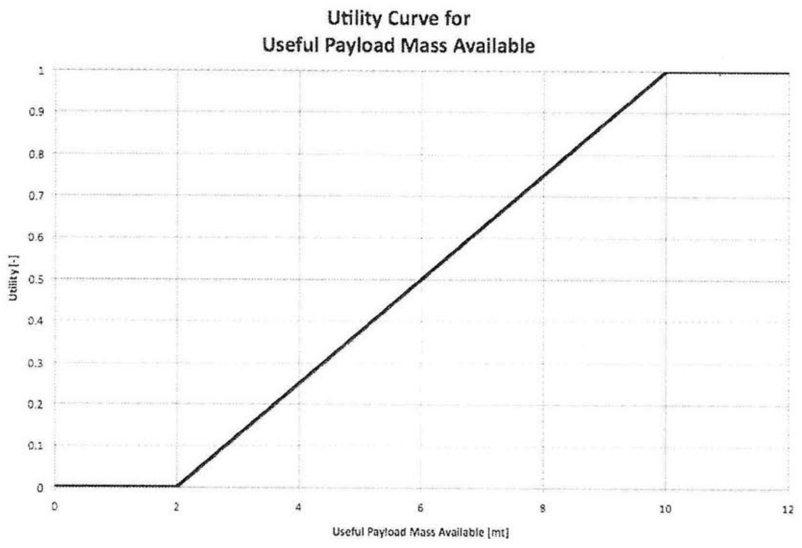


Figure 31. Utility curve for the useful payload mass available [mt].

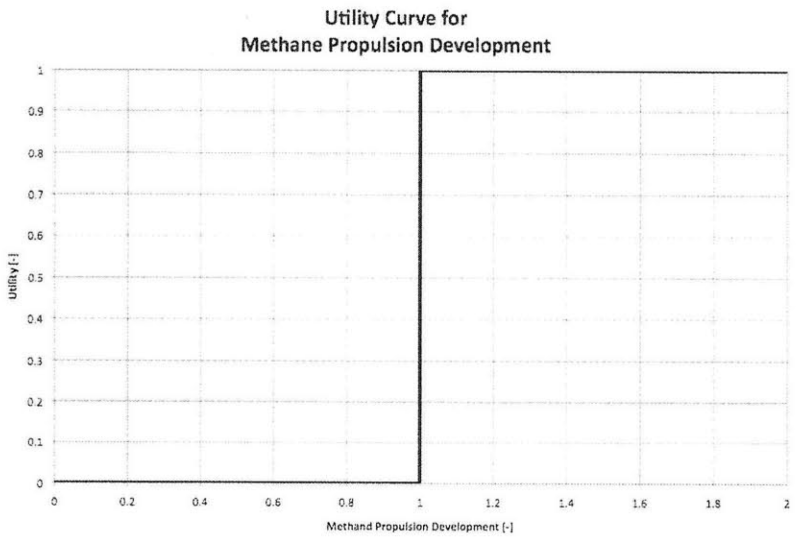


Figure 32. Utility curve for methane propulsion development.

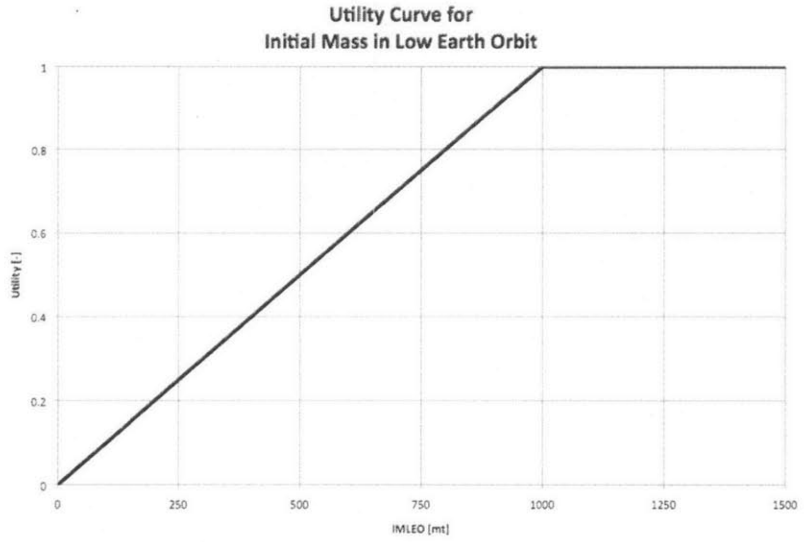


Figure 33. Utility curve for initial mass in LEO [mt].

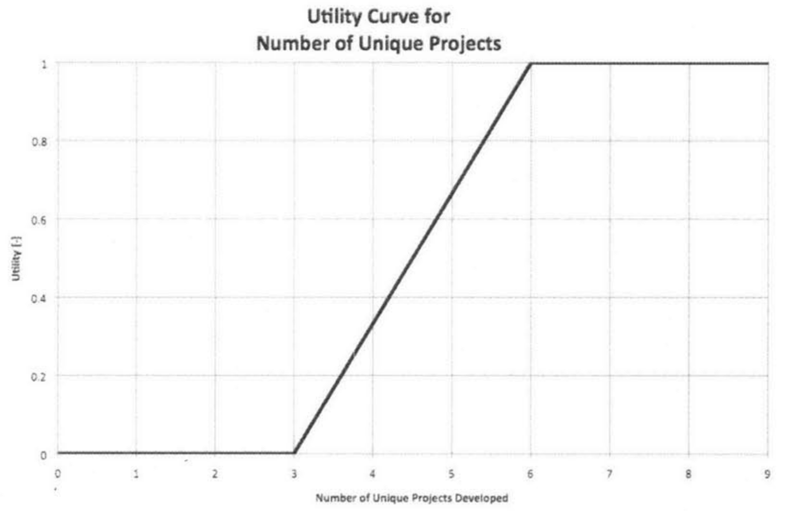


Figure 34. Utility curve for number of unique development projects.

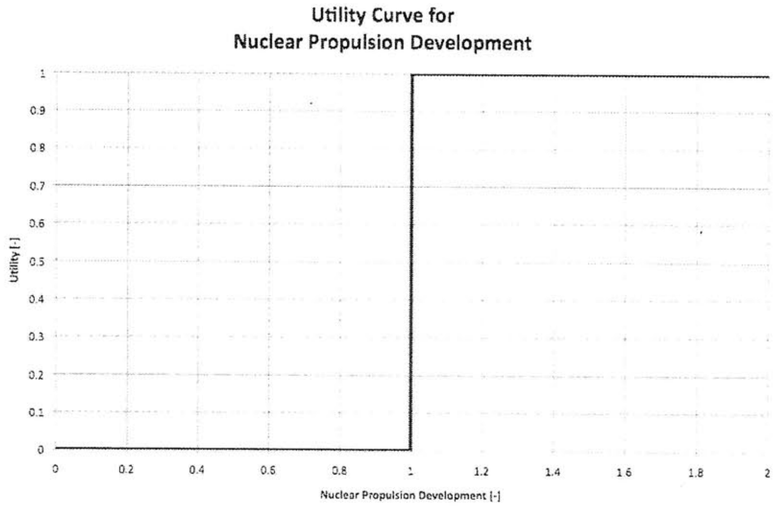


Figure 35. Utility curve for nuclear propulsion development.

Assuming an equal weighting of each property variable, a single metric was developed for the benefit for the system (from the average of the four benefit property variable utilities) and for the cost for the system (from the average of the three cost related property variables utilities). Figure 36 shows a Pareto Front View plotting the benefit versus the cost for the feasible NEO architectures.

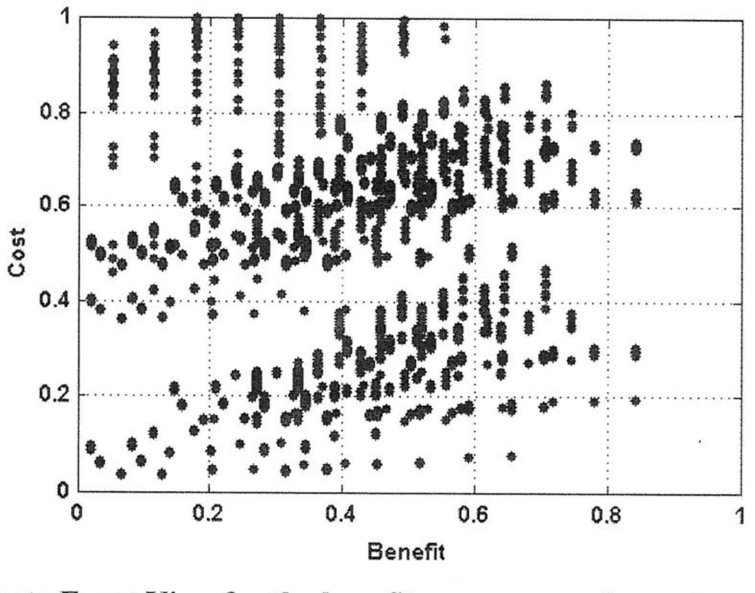


Figure 36. Pareto Front View for the benefit score versus the cost score for feasible NEO architectures.

Table 54 through Table 56 show the top ranked architectures based on highest benefit, lowest cost, and highest value (benefit/cost) for NEO mission architectures. The following insight can be gained from the model based on these results:

- The long duration, low delta-v opportunity for a NEO mission provides both the highest benefit as well as the lowest cost of any mission type.
- Larger crews provide more benefit, however, smaller crews provide lower costs and value is relatively independent from crew size.
- Higher mass for useful payload provides more benefit, however, smaller masses for useful payload provide lower costs and higher overall value.
- Propulsion commonality does not provide high benefit, but it does reduce cost and is present in the highest value architectures.
- Methane propulsion always shows up in the highest benefit architectures and highest value architectures, while NTP never shows up in the lowest cost architectures and highest value architectures.
- The presence of an over-sized re-entry vehicle does not impact the cost or value of the architecture to a significant degree.

Table 54. Top 18 NEO mission architectures based on benefit.

missionType	number of Crew	propCommonality	Useful Payload [mt]	earthDept	hab	destArr	destDept	reentry	nucPropDev	methPropDev	benefit score	cost score	value score
304 day	6	none	10	LOX/ LH2	6	LOX/ LCH4	NTP	6 crew	Y	Y	0.84	0.74	1.14
304 day	6	none	10	LOX/ LH2	6	NTP	LOX/ LCH4	6 crew	Y	Y	0.84	0.73	1.16
304 day	6	none	10	LOX/ LCH4	6	LOX/ LCH4	NTP	6 crew	Y	Y	0.84	0.74	1.14
304 day	6	none	10	LOX/ LCH4	6	NTP	LOX/ LH2	6 crew	Y	Y	0.84	0.73	1.15
304 day	6	none	10	NTP	6	LOX/ LH2	LOX/ LCH4	6 crew	Y	Y	0.84	0.73	1.16
304 day	6	none	10	NTP	6	LOX/ LCH4	LOX/ LH2	6 crew	Y	Y	0.84	0.73	1.15
304 day	6	partial	10	LOX/ LH2	6	LOX/ LH2	LOX/ LCH4	6 crew	N	Y	0.84	0.29	2.93
304 day	6	partial	10	LOX/ LH2	6	LOX/ LCH4	LOX/ LH2	6 crew	N	Y	0.84	0.30	2.85
304 day	6	partial	10	LOX/ LH2	6	LOX/ LCH4	LOX/ LCH4	6 crew	N	Y	0.84	0.30	2.84
304 day	6	partial	10	LOX/ LCH4	6	LOX/ LH2	LOX/ LH2	6 crew	N	Y	0.84	0.29	2.88
304 day	6	partial	10	LOX/ LCH4	6	LOX/ LH2	LOX/ LCH4	6 crew	N	Y	0.84	0.29	2.87
304 day	6	partial	10	LOX/ LCH4	6	LOX/ LCH4	LOX/ LH2	6 crew	N	Y	0.84	0.30	2.78
304 day	6	partial	10	LOX/ LCH4	6	LOX/ LCH4	NTP	6 crew	Y	Y	0.84	0.64	1.33
304 day	6	partial	10	LOX/ LCH4	6	NTP	LOX/ LCH4	6 crew	Y	Y	0.84	0.62	1.36
304 day	6	partial	10	LOX/ LCH4	6	NTP	NTP	6 crew	Y	Y	0.84	0.62	1.36
304 day	6	partial	10	NTP	6	LOX/ LCH4	LOX/ LCH4	6 crew	Y	Y	0.84	0.62	1.35
304 day	6	partial	10	NTP	6	LOX/ LCH4	NTP	6 crew	Y	Y	0.84	0.62	1.35
304 day	6	partial	10	NTP	6	NTP	LOX/ LCH4	6 crew	Y	Y	0.84	0.61	1.38
304 day	6	full	10	LOX/ LCH4	6	LOX/ LCH4	LOX/ LCH4	6 crew	N	Y	0.84	0.19	4.37

Table 55. Top 10 NEO mission architectures based on lowest cost score.

missionType	number of Crew	propCommonality	Useful Payload [mt]	earthDept	hab	destArr	destDept	reentry	nucPropDev	methPropDev	utility score	cost score	value score
304 day	2	full	2	LOX/ LH2	2 per 180 day	LOX/ LH2	LOX/ LH2	2 crew	N	N	0.07	0.03	2.01
304 day	2	full	2	LOX/ LH2	2 per 180 day	LOX/ LH2	LOX/ LH2	4 crew	N	N	0.07	0.03	1.90
304 day	2	full	4	LOX/ LH2	2 per 180 day	LOX/ LH2	LOX/ LH2	2 crew	N	N	0.13	0.03	3.71
304 day	2	full	2	LOX/ LH2	2 per 180 day	LOX/ LH2	LOX/ LH2	6 crew	N	N	0.07	0.04	1.80
304 day	2	full	4	LOX/ LH2	2 per 180 day	LOX/ LH2	LOX/ LH2	4 crew	N	N	0.13	0.04	3.52
304 day	2	full	4	LOX/ LH2	2 per 180 day	LOX/ LH2	LOX/ LH2	6 crew	N	N	0.13	0.04	3.35
304 day	2	full	2	LOX/ LCH4	2 per 180 day	LOX/ LCH4	LOX/ LCH4	2 crew	N	Y	0.32	0.04	7.63
304 day	2	full	2	LOX/ LCH4	2 per 180 day	LOX/ LCH4	LOX/ LCH4	4 crew	N	Y	0.32	0.04	7.22
304 day	2	full	4	LOX/ LCH4	2 per 180 day	LOX/ LCH4	LOX/ LCH4	2 crew	N	Y	0.38	0.04	8.65
304 day	4	full	2	LOX/ LH2	4 per 180 day	LOX/ LH2	LOX/ LH2	4 crew	N	N	0.20	0.04	4.56

Table 56. Top ten ranked architectures in terms of value (utility / cost).

missionType	# crew	Prop Commonality	Payload [mt]	earthDept	hab	destArr	destDept	reentry	nucPropDev	methPropDev	utility score	cost score	value score
304 day	6	full	4	LOX/ LCH4	6 crew 180 day	LOX/ LCH4	LOX/ LCH4	6 crew	N	Y	0.65	0.07	8.80
304 day	4	full	4	LOX/ LCH4	4 crew 180 day	LOX/ LCH4	LOX/ LCH4	4 crew	N	Y	0.52	0.06	8.75
304 day	2	full	4	LOX/ LCH4	2 crew 180 day	LOX/ LCH4	LOX/ LCH4	2 crew	N	Y	0.38	0.04	8.65
304 day	4	full	4	LOX/ LCH4	4 crew 180 day	LOX/ LCH4	LOX/ LCH4	6 crew	N	Y	0.52	0.06	8.41
304 day	6	full	2	LOX/ LCH4	6 crew 180 day	LOX/ LCH4	LOX/ LCH4	6 crew	N	Y	0.59	0.07	8.22
304 day	2	full	4	LOX/ LCH4	2 crew 180 day	LOX/ LCH4	LOX/ LCH4	4 crew	N	Y	0.38	0.05	8.21
304 day	4	full	2	LOX/ LCH4	4 crew 180 day	LOX/ LCH4	LOX/ LCH4	4 crew	N	Y	0.45	0.06	8.01
304 day	2	full	4	LOX/ LCH4	2 crew 180 day	LOX/ LCH4	LOX/ LCH4	6 crew	N	Y	0.38	0.05	7.81
304 day	4	full	2	LOX/ LCH4	4 crew 180 day	LOX/ LCH4	LOX/ LCH4	6 crew	N	Y	0.45	0.06	7.69
304 day	2	full	2	LOX/ LCH4	2 crew 180 day	LOX/ LCH4	LOX/ LCH4	2 crew	N	Y	0.32	0.04	7.63

4.5.2 Determining the overall influence of a decision

The second type of information that can be viewed from a decision-based system architecture model is the information related to how influential a decision is to a system. Figure 37 through Figure 43 show the individual Decision Space Views for each of the property variables of interest. Table 57 and Table 58 provide additional information in the form of the entries for the individual and average Property Value Sensitivities for each decision variable as well as the calculation of the overall Influence Score that was introduced in Chapter Two. All data was calculated for the pareto set of architectures.

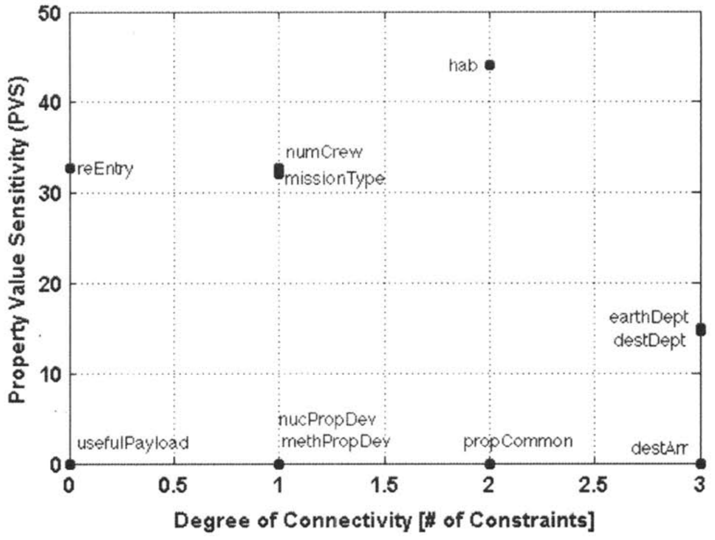


Figure 37. Decision Space View for number of crew-days at the NEO.

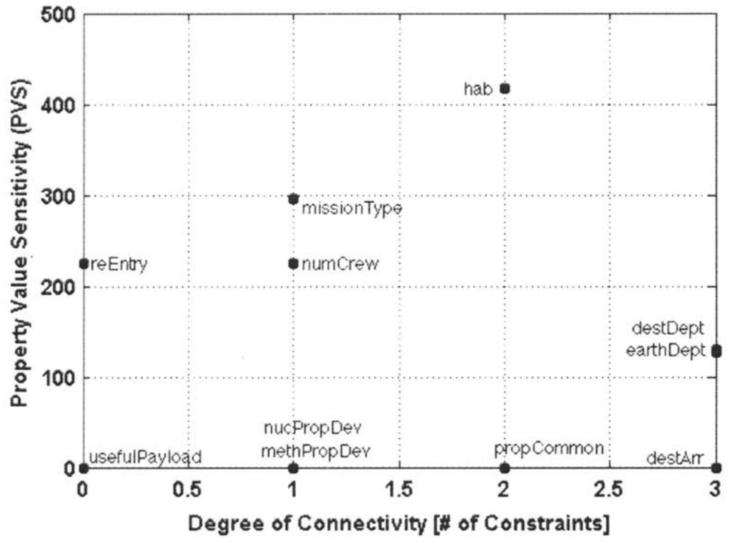


Figure 38. Decision Space View for number of crew-days in space.

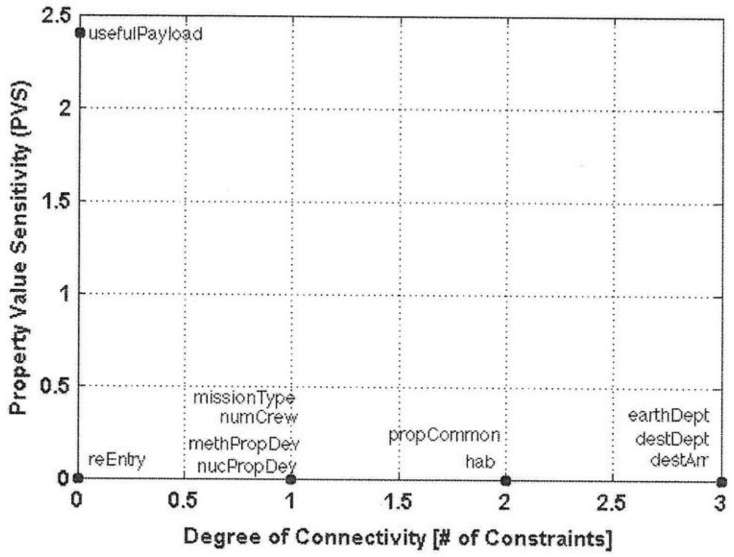


Figure 39. Decision Space View for useful payload mass.

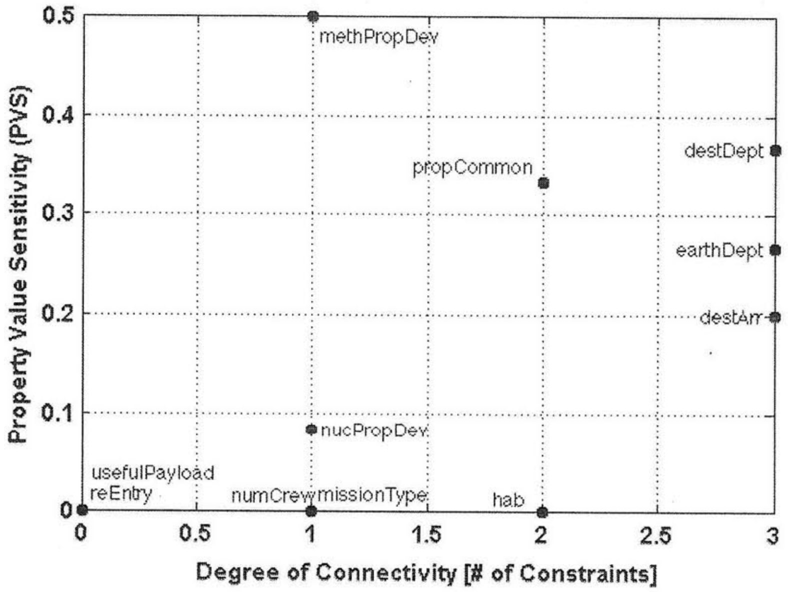


Figure 40. Decision Space View for methane propulsion development.

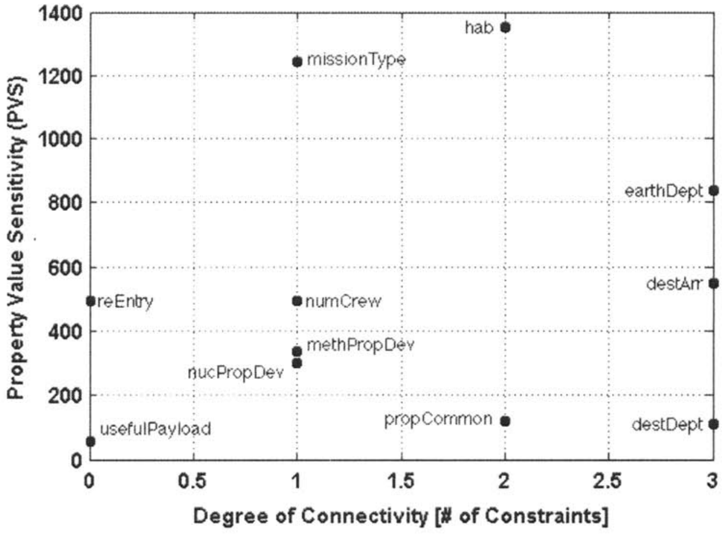


Figure 41. Decision Space View for IMLEO.

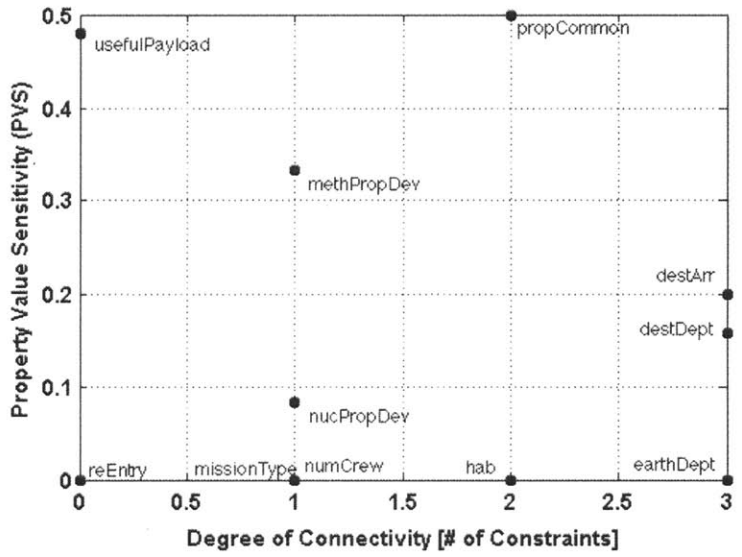


Figure 42. Decision Space View for number of unique development projects.

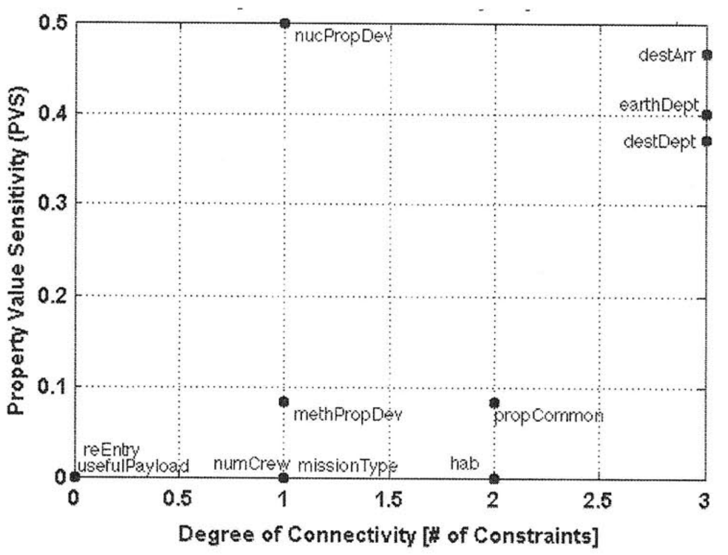


Figure 43. Decision Space View for the development of nuclear propulsion.

Table 57. Calculation of average Property Value Sensitivities for NEO mission-level model.

	Property Value Sensitivity										
	crewdays at destination	crewdays space	useful payload	methane devleopment?	IMLEO	# unique projects	nuclear development?	Benefit PVS	Cost PVS	Utility PVS	Degree of connectivity
destDept	0.12	1.00	0.00	0.00	0.86	0.00	0.00	0.28	0.29	0.28	3
number of Crew	0.07	0.45	0.00	0.00	1.00	0.00	0.00	0.13	0.33	0.23	1
reentry	0.07	0.45	0.00	0.00	1.00	0.00	0.00	0.13	0.33	0.23	0
hab	0.03	0.31	0.00	0.00	1.00	0.00	0.00	0.09	0.33	0.21	2
missionType	0.03	0.24	0.00	0.00	1.00	0.00	0.00	0.07	0.33	0.20	1
earthDept	0.02	0.15	0.00	0.00	1.00	0.00	0.00	0.04	0.33	0.19	3
Useful Payload	0.00	0.00	0.04	0.00	1.00	0.01	0.00	0.01	0.34	0.17	0
propCommonality	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.33	0.17	2
methPropDev	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.33	0.17	1
nucPropDev	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.33	0.17	1
destArr	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.33	0.17	3

Table 58. Calculation of the Average Influence Score for the NEO mission-level model.

	Influence Score										
	crewdays at destination	crewdays space	useful payload	methane devleopment?	IMLEO	# unique projects	nuclear development?	Benefit AIS	Cost AIS	Total AIS	Degree of connectivity
destDept	0.56	1.00	0.50	0.50	0.93	0.50	0.50	0.64	0.64	0.64	3
earthDept	0.51	0.58	0.50	0.50	1.00	0.50	0.50	0.52	0.67	0.59	3
destArr	0.50	0.50	0.50	0.50	1.00	0.50	0.50	0.50	0.67	0.58	3
hab	0.35	0.49	0.33	0.33	0.83	0.33	0.33	0.38	0.50	0.44	2
propCommonality	0.33	0.33	0.33	0.33	0.83	0.34	0.33	0.33	0.50	0.42	2
number of Crew	0.20	0.39	0.17	0.17	0.67	0.17	0.17	0.23	0.33	0.28	1
missionType	0.18	0.29	0.17	0.17	0.67	0.17	0.17	0.20	0.33	0.27	1
methPropDev	0.17	0.17	0.17	0.17	0.67	0.17	0.17	0.17	0.33	0.25	1
nucPropDev	0.17	0.17	0.17	0.17	0.67	0.17	0.17	0.17	0.33	0.25	1
reentry	0.03	0.23	0.00	0.00	0.50	0.00	0.00	0.06	0.17	0.12	0
Useful Payload	0.00	0.00	0.02	0.00	0.50	0.00	0.00	0.01	0.17	0.09	0

Based on the above information, the influence of the three propulsion element selection decisions, particularly the propulsion system for departing from the NEO (as it must be carried throughout the mission) can be seen along with the influence of the decision to enforce commonality among the propulsion elements. It should be noted that the apparent influence of the choice of the re-entry vehicle is directly linked to the fact that the re-entry vehicle choice changes in step with the number of crew for the pareto set so any sensitivity to the number of crew a property variable has, it has the exact same sensitivity to the choice of re-entry vehicle for the pareto set.

4.6 Summary

This chapter described the details of applying a single-level decision-based system architecture model to the investigation of system architectures for a human NEO mission. Section 4.2 showed how the trade space for a human NEO mission could be represented through 7 property variables and functions, 11 decision variables, and 6 logical constraints. Section 4.3 and 4.4 outlined how structural reasoning and simulation were applied to the model. Section 4.5 described the data that was collected from the single-level decision-based model and provide insight into trends among interesting architectures as well as insight into which decision variables are the most important throughout the system.

The next chapter discusses the theory behind multi-level decision-based system architecture modeling. This prepares the reader for Chapter Six, which examines the NEO mission architectures further by investigating the habitation element in more detail.

5 A Framework for Multi-Level Decision-Based System Architecture Modeling

5.1 Introduction

Decision-based system architecture modeling represents the system architecting process for a complex system as a set of interconnected decisions that can be made about the system and its components and evaluates the feasible combinations of these decisions (i.e. feasible architectures) using a set of property variables based on stakeholder values. Earlier chapters discussed the importance of the principle of levels of system abstraction on decision-based modeling. By ensuring that all the decisions in a given model deal with the same level of abstraction for the system, validity can be maintained (but not guaranteed) for the decision-based viewing tools, which are what differentiates decision-based modeling from other types of trade space exploration and optimization frameworks.

The negative effect associated with requiring that all decisions included in a model are on the same level of abstraction is that, if the system is modeled at lower levels of abstraction (e.g. Level 2 or lower), the model for the entire system becomes unmanageable both in terms of computational requirements and human comprehension, however, if the system is modeled at a high level (e.g. Level 1) as done in the previous chapters, the model does not give insight into lower level decisions related to subsystem and component design. Devising a framework that allows investigation of lower-level (e.g. Level 2) decisions while maintaining validity of the decision-based viewing tools as well as the ability for human comprehension of the model would provide a system architect with further insight into the architectural trade space and its related decisions.

The objective of this chapter is to discuss the theory behind the development of a multi-level decision-based system architecting framework. Section 5.2 outlines the principles behind the development of a multi-level framework. Section 5.3 details the three-step process of developing a multi-level model. Section 5.4 provides discussion about the validity and efficiency of the proposed multi-level model. The chapter is summarized in Section 5.5.

5.2 A Multi-Level Decision-Based System Architecting Framework

As discussed in Chapter Two, a single-level decision-based model enumerates and evaluates feasible combinations of decision variables of a system at a specified level of abstraction of the architecture (e.g. Level 1). Complex systems are composed of elements at Level 1 that can often be considered as complex systems in and of themselves. Therefore, it is feasible to develop single-level decision-based models to enumerate and evaluate the architectures for each element of a complex system by thinking of the element as a system itself. Developing single-level models for each individual element (i.e. a Level 2 model) allows an exploration of the trade space related to that specific element.

The concept of a multi-level decision-based framework is that this hierarchical sequence of single-level models can be linked together through the variables, relations, and characteristics of the applicable models. The feasible architectures created for each element (i.e. the architectures produced through the Level 2 models) are used as the alternatives of the element available for evaluation in the design of the parent system (i.e. at Level 1). Specifically, the non-dominated architectures developed at the lower-level become the decision alternatives for the related form-function mapping decisions at the higher level.

The challenge in multi-level decision-based system architecting is ensuring that 1) the Level 2 model provides all the information required by the Level 1 and 2) the architectures passed from Level 2 to Level 1 include all interesting options when viewed at the higher level. The solution to both these challenges is reached by ensuring that the property variables for the Level 2 model are selected appropriately.

As described in Chapter Two, a single-level model evaluates each of the feasible architectures by using the characteristics of the selected alternatives to calculate the property functions and property variables. Therefore, since an architecture created by a Level 2 model becomes an alternative for the related decision in the Level 1 model, the Level 2 model must provide the characteristics required by the Level 1 model. This is accomplished by setting a property variable in the Level 2 model to the relevant characteristics at Level 1. Knowledge of the Level 1 model is necessary in order to know if the Level 1 characteristic should be maximized, minimized, or simply recorded in order to property set the corresponding Level 2 property variable. Since the non-dominant designs from the Level 2 model populate the alternatives in the Level 1 model, if it is known that only element designs with either a maximized or minimized characteristic are preferred, then only the architectures that meet that criteria will be passed to the higher-level model.

If the Level 2 model only had property variables related to characteristics required for the element in the Level 1 model, there would be no way of ensuring that all preferred Level 2 architectures are passed since there would be no way to account for the interactions between the elements at Level 1. In decision-based modeling, these interactions are captured through the logical constraints. Each decision in a single-level model may be connected to other decisions through one or more logical constraints. These logical constraints limit the feasible alternatives for a given decision based on the alternatives selected for other decisions. Property variables must be included in the Level 2 model in order to account for these Level 1 logical constraints and to ensure that the set of dominated Level 2 architectures is populated with knowledge of the Level 1 interactions.

Just as it was shown in Chapter Two how property functions are related to the decision alternatives through characteristics of the alternatives, it is also true that logical constraints limit the feasible alternatives for a decision based on the characteristics of the alternatives. In any given architecture, if a given decision is set to a certain alternative, an alternative for another decision may or may not be feasible if it has a certain characteristic that is less than, greater than, or equal to some specified value. It is this characteristic of the alternatives that is linked to Level 1 logical constraints that must be set as a property variable at Level 2. This allows the Level 1 interactions to be captured in the Level 2 model. It is through knowledge of the Level 1 model that the Level 2 property variable is set to be minimized, maximized, or simply counted.

This concept of dividing a mission into a set of lower-level systems consisting of the elements and subsystems that must be designed mirrors the real-world design practice of many human spaceflight organizations as shown in Figure 44. For the CxAT studies on the design of the Lunar Surface System [Coo08], NASA had several levels of design teams. Lower-level design teams would focus on designing specific subsystems based on the requirements for the higher-level elements. The element teams would then use a combination of the proposed designs for each of their relevant subsystems and examine the trade space of options for the overall element based on the requirements given to them by a mission-level team and provide a set of concepts for their element to that higher-level team so that they, in turn, could investigate the preferred set of elements to complete the mission. By developing a multi-level decision-based framework that mimics the real-world design practices of the industry, it will be easier for a system architect to discuss the specific details of the modeling process with relevant stakeholders.

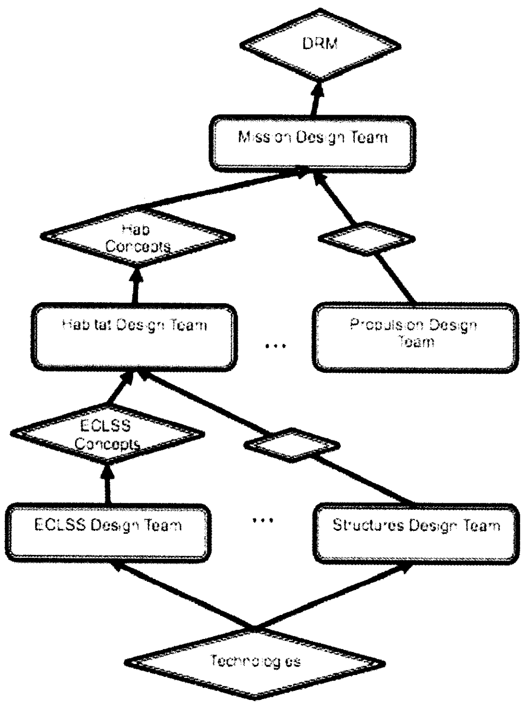


Figure 44. Organization of design teams within human spaceflight organizations.

The remaining sections in this chapter focus on the steps required in implementing a multi-level decision-based model for a human spaceflight project. The theory will be complemented with a continuation of the example problem from Chapter Two.

5.3 The Multi-Level Decision-Based Modeling Process

The main concept behind multi-level decision-based modeling is to develop a series of hierarchical single-level models. Instead of just having one model for the Level 1 decisions, additional models are created for each element that the system architect desires further insight into (i.e. Level 2 model). The multi-level modeling process can be thought of as three steps that should be run in order and then iterated upon as necessary. These steps are shown in Table 59.

Table 59. Steps involved in the multi-level decision-based modeling process.

Step 1.	Develop the mission-level model using single-level decision-based architecture approach.
Step 2.	Develop and run element-level models for relevant elements.
Step 3.	Inform and re-run the mission-level model with new element alternatives.

The following subsections discuss the details related to each step in the process and illustrate the step using the example problem discussed in Chapter Two.

5.3.1 Step 1: Develop the mission-level model

The first step in developing a multi-level framework is to develop a single-level model for the mission-level (or Level 1) architecture. It is necessary to develop a mission-level model before investigating element-level (or Level 2) models because without an understanding of the mission-level concept, it is impossible to design appropriate elements. The process for developing a single-level mission level model is discussed in Chapter Two. At this early stage of modeling, the system architect should be able to discern the system attribute decisions and alternatives for the mission-level as well as the set of decision variables related to the internal functions required by the mission along with the property variables and functions. Populating the alternatives for the form-function mapping decisions is done using existing designs from literature or rough concepts based on previous experience. These alternatives will be refined later in the multi-level decision-based modeling process.

The exact tasks involved in developing the mission-level model for the example problem can be seen in Chapter Two. Table 60 to Table 63 summarize the representation of this model. Figure 45 shows the Pareto Front View for the mission-level model, while Table 65 shows the calculation of the Average Influence Score for the mission-level model.

Table 60. Decision variables for the example problem.

Decision	Alt A	Alt B	Alt C
#crew: How many crew are there?	1	2	
duration: What is the duration of the mission?	7 days	14 days	
habitat: How is a habitable volume provided by the system?	Habitat 1.7 (1crew & 7crewday)	Habitat 2.14 (2crew & 14crewday)	Habitat 2.28 (2crew & 28crewday)
power: How is power provided by the system?	7-day battery	14-day battery	

Table 61. List of logical constraints for the example problem.

Constraint Name	Decision Variables Affected	Constraint Statement
Constraint 1	# crew, duration	if (# crew = A), then (duration = A|B), elseif (#crew = B), then (duration = B)
Constraint 2	#crew, habitat	if (#crew = A), then (habitat = A|B|C|D), elseif (#crew = B), then (habitat = C|D)
Constraint 3	#crew, duration, habitat	if (#crew = A & duration = A), then (habitat = A|B|C), elseif (#crew = A & duration = B), then (habitat = B|C), elseif (#crew = B & duration = A), then (habitat = B|C), elseif (#crew = B & duration = B), then (habitat = C)
Constraint 4	duration, power	if (duration = A), then (power = A), elseif (duration = B), then (power = B)

Table 62. Property functions for the example problem.

Property Variable	Property Function
Mass of System	sum(element masses)
Number of crew-days	Number of crew x Mission duration

Table 63. Characteristics for the example problem decision alternatives.

Decision Variables	Alternatives	Characteristics		
		Mass [mt]	Number of Crew [#]	Mission Duration [days]
#crew	1	-	1	-
	2	-	2	-
duration	7	-	-	7
	14	-	-	14
habitat	Habitat 1.7	1.3	-	-
	Habitat 2.14	2.6	-	-
	Habitat 2.28	2.9	-	-
power	7-day battery	2	-	-
	14-day battery	4	-	-

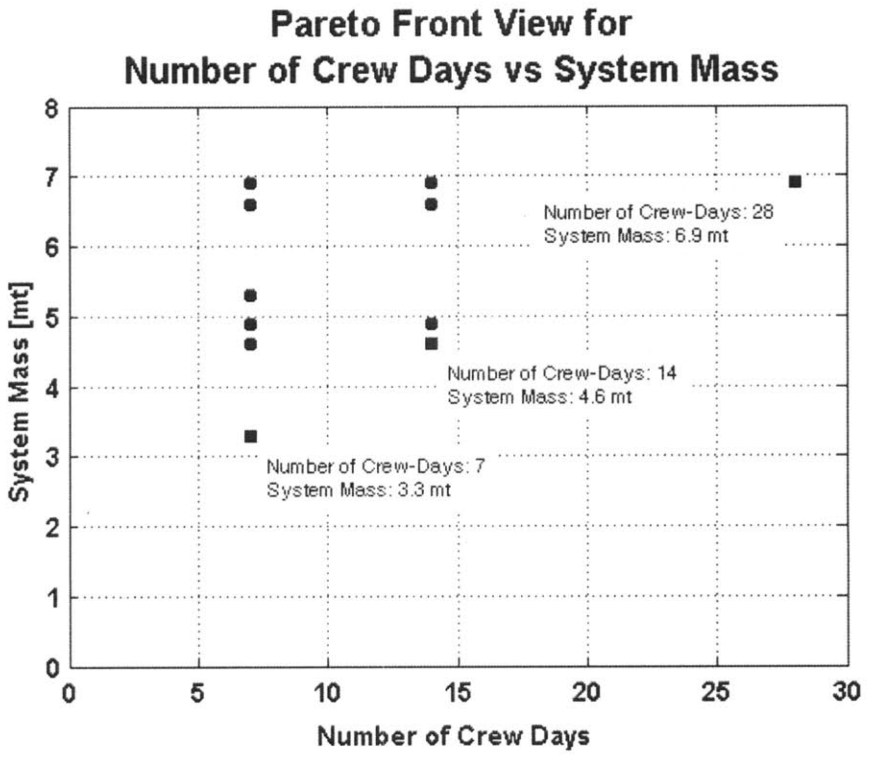


Figure 45. Pareto Front View for the mission level of the example problem.

Table 64. Property variables values and decision selections for the dominant architectures.

Arch #	Number of Crew	Mission Duration [days]	Habitat	Power System	Crew-days supported	System Mass [mt]
1	1	7	Habitat 1.7	7day battery	7	3.3
7	2	7	Habitat 2.14	7day battery	14	4.6
11	2	14	Habitat 2.28	14day battery	28	6.9

Table 65. AIS calculation for the example problem.

	# of crew-days		Benefit AIS	Mass [mt]		Cost AIS	Value AIS
	Connectivity	PVS		Connectivity	PVS		
numCrew	3	7	0.66	3	1.23	0.69	0.67
duration	3	21.43	1.00	3	3.32	1.00	1.00
habitat	2	13.23	0.64	2	2.22	0.67	0.65
power	1	21.43	0.67	1	3.32	0.67	0.67

Once the mission-level model has been developed, it must be determined which elements are of interest in terms of further investigation. For this problem, it is assumed that the habitat is of interest and the stakeholders desire further definition. It is important to note that not all elements have to have their own element-level model; the multi-level framework allows the depth of the investigation of the element to be tailored to the importance of each individual element and subsystem.

5.3.2 Develop the element-level models of interest

The second step in the multi-level decision-based modeling framework is to develop and run the element-level (i.e. Level 2) models of interest. The representation of this level of model will be informed and impacted by the representation of the Level 1 mission model. The key to multi-level modeling is ensuring that the property variables of the element-level models are chosen to enable linkage to the higher-level models through the relevant characteristics and logical constraints. In the case of the example problem, it is the habitat for which an element-level model is desired.

5.3.2.1 Choosing the property variables

The first step in representing the habitat system as a decision-based model is choosing the property variables. From Table 63, it can be seen that the Level 1 decision for the habitat element has a single characteristic that is required for that model: the element mass. From Table 61, it can be seen that the Level 1 decision is connected to other decisions through two logical constraints. Constraint #2 states that the habitat must be able to support, at a minimum, the number of crew chosen for the mission concept. Through this Level 1 logical constraint, it is implied that each habitat alternative has a characteristic of 'the maximum number of crew the habitat can support'. Therefore, it is of interest at the element-level (i.e. Level 2) to include a property variable related to the maximum number of crew that the element can support. Constraint #3 states that the habitat must be able to support, at a minimum, the number of crew-days related to the mission. It is implied that the habitat alternatives have a characteristic of 'the maximum number of crew-days the habitat can support'. Therefore, another element-level property variable should be related to the maximum number of crew-days that the element can support.

Through the development and simulation of the Level 1 model, the architect should be able to determine that:

- The lower the element masses at Level 1, the lower the overall mass of the system.
- The more crew-members that a mission can support (and therefore a habitat can support, the more crew-days a mission can have which is a Level 1 property variable.
- With more crew-days on a mission, more benefit can be obtained. Therefore, the more crew-days a habitat can support, the better.

Therefore, the investigation of the mission-level (Level 1) model has led the element-level (Level 2) model for the habitat to have the following three property variables:

- Element mass (lower is better)
- Maximum number of crew supported (higher is better)
- Maximum number of crew-days supported (higher is better)

5.3.2.2 Choosing the decision variables

Once the property variables have been chosen, the decision variables must be developed. Following the process of developing a single-level model, both system attribute decision variables and form-to-function mapping decision variables can be developed. Based on the three property variables chosen, the decisions variables must be able to capture both the maximum number of crew and the maximum number of crew-days (i.e. number of crew multiplied by mission duration) as well as the overall mass of the habitat. In the case of the example problem, the system attribute decisions at the element-level are:

- What is the maximum number of crewmembers the habitat must support? (1 or 2)

- What is the maximum number of days that the habitat must operate in space? (7 or 14)

The similarity of these system parameter decisions and alternatives and their mission-level counterparts ensures that the attribute options investigated for the architectures of the habitat element coincide with the attribute options investigated for the overall system (at Level 1).

The form-function mapping decisions in the element model should focus on the internal functions that the habitat must perform to support the mission. For the example problem, these functions include: providing enough pressurized volume for the number of crew present and providing a life support system capable of support the number of crew-days required by the mission. The alternatives for these decisions are feasible subsystem designs that can perform the function in question. Since the system architect for the habitation model, and not the architect for the mission, does this model, the trade space explored is larger because the architect has more resources available and expertise in investigating the habitation design compared to the mission-level architects. The additional alternatives (designated by the choices with 1b, 2b, 14b, or 28b) represent additional subsystem designs. For simplicity, their capabilities are similar to their counterparts used in the original Level 1 model (designated by 1a, 2a, 7a, 14a, or 28a), but they have different masses. For example, the habitat design in the Level 1 model listed as Habitat 1.7 is the same as the Habitat 1a.7a as developed by the Level 2 model. The decision variables and the simplified set of feasible subsystem designs are shown in Table 66.

Table 66. Form-function decision variables for the habitat element model.

Decision Variable	Alt A	Alt B	Alt C	Alt D	Alt E
How is the pressurized volume provided?	Structural Subsystem 1a (1 crew)	Structural Subsystem 1b (1 crew)	Structural Subsystem 2a (2 crew)	Structural Subsystem 2b (2 crew)	
How is the ECLSS functionality provided?	ECLSS Subsystem 7a (7crewdays)	ECLSS Subsystem 14a (14crewdays)	ECLSS Subsystem 14b (14crewdays)	ECLSS Subsystem 28a (28crewdays)	ECLSS Subsystem 28b (28crewdays)

5.3.2.3 Property Functions and Logical Constraints

The property functions for each property variable are shown in Table 67. The necessary characteristics are shown in Table 68. The logical constraints for the habitation element model are shown in Table 69.

Table 67. Level 2 property functions for the example problem.

Property Variable	Property Function
Mass of Element	sum(subsystem masses)
Max Number of Crew	Max Number of Crew
Max Number of Crew-Days	Max Number of crew x Max Mission duration

Table 68. Level 2 model characteristics for the example problem.

Decision Variables	Alternatives	Characteristics		
		Subsystem Mass [mt]	Max # crew	Max Mission Duration
max # crew	1	-	1	-
	2	-	2	-
max duration	7	-	-	7
	14	-	-	14
Structure	Structure 1a	1	-	-
	Structure 1b	1.2	-	-
	Structure 2a	2	-	-
	Structure 2b	1.8	-	-
ECLSS	ECLSS 7a	0.3	-	-
	ECLSS 14a	0.6		
	ECLSS 14b	0.5		
	ECLSS 28a	0.9		
	ECLSS 28b	1.1		

Table 69. Level 2 logical constraints for the example problem.

Constraint Name	Decision Variables Affected	Constraint Statement
Constraint 1	max # crew, structure	if (# crew = A), then structure = A|B|C|D), elseif (#crew = B), then structure = C|D)
Constraint 2	max # crew, max duration, ECLSS	if (#crew = A & duration = A), then ECLSS = A|B|C|D|E), elseif (#crew = A & duration = B), then (ECLSS = B|C|D|E) elseif (#crew = B & duration = A), then (ECLSS = B|C|D|E) elseif (#crew = B & duration = B), then (habitat = E)

5.3.2.4 Structural Reasoning and Simulation

Applying structural reasoning and simulating this model results in 32 feasible architecture choices for the habitat and 6 non-dominated designs based on the property variables. The next step is to take these non-dominated architectures and input them into the mission-level model.

Table 70. Values of the non-dominated habitat designs.

Structure	ECLSS	Mass [mt]	Crew-Days Supported	Crew Supported
1a	7a	1.3	7	1
1a	14b	1.5	14	1
1a	28a	1.9	28	1
2b	7a	2.1	7	2
2b	14b	2.3	14	2
2b	28a	2.7	28	2

5.3.3 Re-running the mission model as informed by the element-model

The final step of the multi-level decision-based modeling process is to inform and re-run the mission-level (Level 1) model using the information gained through the development of the lower-level model (Level 2). To accomplish this step, the dominant designs are taken from the lower-level element models along with their property variables and these designs become the alternatives for the related form-function mapping decision at the mission-level. The property variables for the lower-level architectures become the new characteristics for the alternatives necessary for calculating the Level 1 property variables. Additionally, the logical constraints have to be rewritten to utilize the new, informed characteristics of each alternative related to them.

In the Level 1 model for the example problem, the habitat element decision is connected to the number of crew decision through one constraint and to the number of crew and mission duration (both used to calculate the crew-days) through another constraint. These logical constraints are rewritten as logical constraint functions using the appropriate property variables from the Level 2 model. For example, the logical constraint connecting the habitat to the number of crew would now take the form of ensuring that the crew supported characteristic of the habitat design is greater or equal to the alternative chosen for the number of crew decision.

Table 71 through

Decision Variables	Alternatives	Characteristics				
		Mass [mt]	Max Number of Crew Supported [-]	Max Number of Crew-Days Supported [-]	Number of Crew on Mission [#]	Mission Duration [days]
#crew	1	-	-	-	1	-
	2	-	-	-	2	-
duration	7	-	-	-	-	7
	14	-	-	-	-	14
habitat	Habitat 1a.7a	1.3	1	7	-	-
	Habitat 1a.14b	1.5	1	14	-	-
	Habitat 1a.28a	1.9	1	28	-	-
	Habitat 2b.7a	2.1	2	7	-	-
	Habitat 2b.14b	2.3	2	14	-	-
	Habitat 2b.28a	2.7	2	28	-	-
power	7-day battery	2	-	-	-	-
	14-day battery	4	-	-	-	-

Table 74 summarize the informed representation of the mission-level model. The habitat alternatives are called out by their structure subsystem and their ECLSS subsystem classifications. It should be noted that there are now habitat alternatives capable of supporting 1 crew for 14 days and 28 days and of supporting 2 crew for 7 days. These alternatives were not enumerated in the original Level 1 model, but have been created by feasible (if not preferred) combinations of subsystem components through the Level 2 model. Table 74 shows the logical constraints for the informed Level 1 model including the re-written constraints based on the characteristics of the habitation element.

Table 71. Decision variables and alternatives for the informed mission-level model.

Decision Variables	Alt A	Alt B	Alt C	Alt D	Alt E	Alt F
#crew: How many crew are there?	1	2				
duration: What is the duration of the mission?	7 days	14 days				
habitat: How is a habitable volume provided by the system?	Habitat 1a.7a (1crew & 7crewday)	Habitat 1a.14b (1crew & 14crewday)	Habitat 1a.28a (1crew & 28crewday)	Habitat 2b.7a (2crew & 7crewday)	Habitat 2b.14b (2crew & 14crewday)	Habitat 2b.28a (2crew & 28crewday)
power: How is power provided by the system?	7-day battery	14-day battery				

Table 72. Property variables for the informed mission-level model.

Property Variable	Property Function
Mass of System	sum(element masses)
Number of crew-days	Number of crew x Mission duration

Table 73. Characteristics for the informed mission-level model.

Decision Variables	Alternatives	Characteristics				
		Mass [mt]	Max Number of Crew Supported [-]	Max Number of Crew-Days Supported [-]	Number of Crew on Mission [#]	Mission Duration [days]
#crew	1	-	-	-	1	-
	2	-	-	-	2	-
duration	7	-	-	-	-	7
	14	-	-	-	-	14
habitat	Habitat 1a.7a	1.3	1	7	-	-
	Habitat 1a.14b	1.5	1	14	-	-
	Habitat 1a.28a	1.9	1	28	-	-
	Habitat 2b.7a	2.1	2	7	-	-
	Habitat 2b.14b	2.3	2	14	-	-
	Habitat 2b.28a	2.7	2	28	-	-
power	7-day battery	2	-	-	-	-
	14-day battery	4	-	-	-	-

Table 74. Logical constraints for the informed mission-level model.

Constraint Name	Decision Variables Affected	Constraint Statement
Constraint 1	duration, power	if (duration = A), then (power = A), elseif (duration = B), then (power = B)
Constraint 2	#crew, habitat	if (#crew = A), then (habitat_crew_characteristic => 1) elseif (#crew = B), then (habitat_crew_characteristic => 2)
Constraint 3	#crew, duration, habitat	if (#crew = A & duration = A), then (habitat_crewday_char >=7) elseif (#crew = A & duration = B), then (habitat_crewday_char >=14), elseif (#crew = B & duration = A), then (habitat_crewday_char >=14), elseif (#crew = B & duration = B), then(habitat_crewday_char >=28)
Constraint 4	# crew, duration	if (# crew = A), then (duration = A|B), elseif (#crew = B), then (duration = B)

With the new alternatives inputted and the revised logical constraints developed, the new, informed mission-level model can be simulated and data related to the entire system based on more information options for the element-level model can be generated. Figure 46 shows the Pareto Front View for the two property variables. The number of feasible architectures in the informed mission-level rose to 37 architectures from 11 architecture while there were still only 3 non-dominated architectures for this example problem with the informed model, in addition, the values of the system masses for the various architectures have changed since different habitats were selected compared to the initial mission-level model. Figure 47 and Figure 48 show the Decision Space Views for the two property variables of interest at the mission-level.

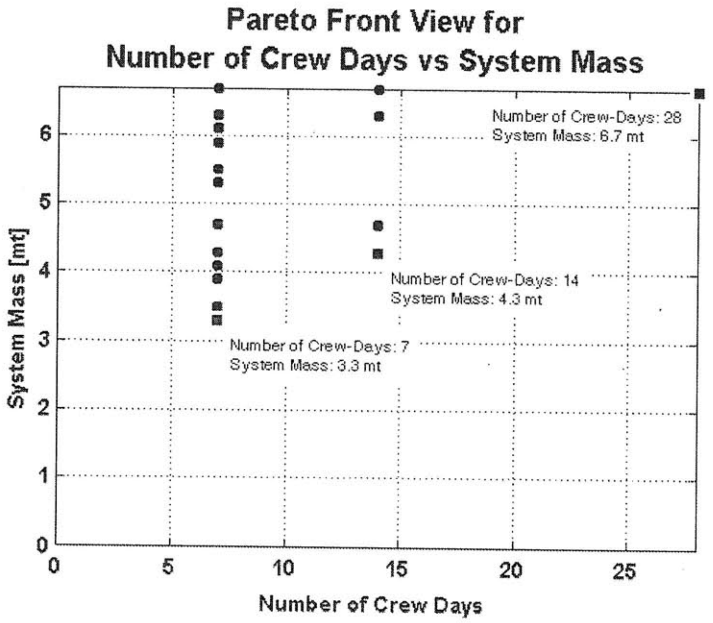


Figure 46. Pareto Front View for Number of Crew-Days vs System Mass for the informed mission-level model.

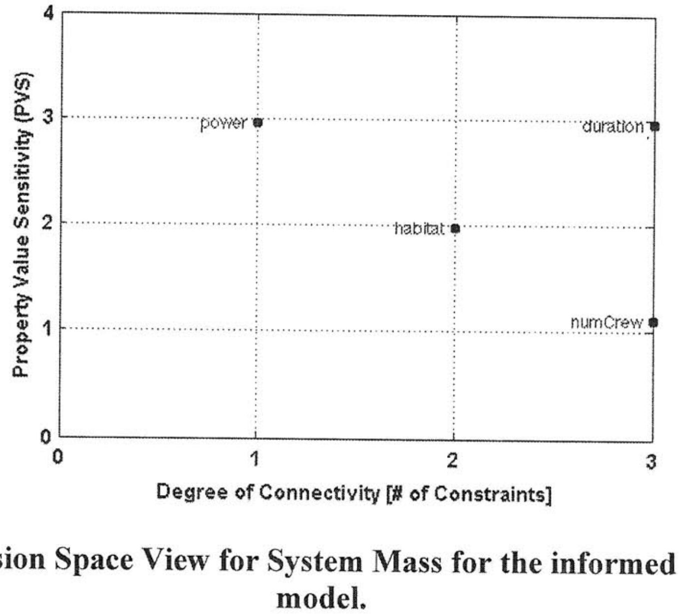


Figure 47. Decision Space View for System Mass for the informed mission-level model.

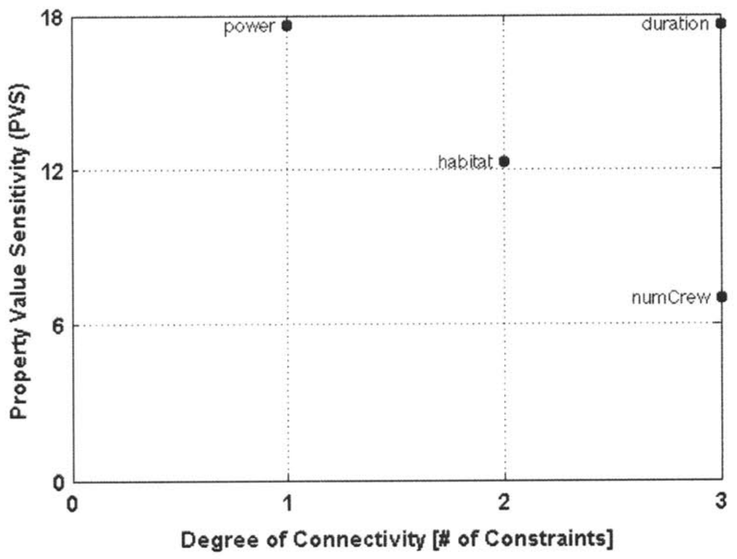


Figure 48. Decision Space View for the Number of Crew-Days for the informed mission-level model.

5.4 Discussion

The multi-level decision-based system architecture modeling framework described above enables a system architect or several different design teams to work together in a cohesive fashion to investigate the architecture trade space and the influence of the relevant decisions for varying levels of abstraction within the overall system. The benefits of this framework for the system architect include:

- Ensuring the validity of the decision-based viewing tools for each level of abstraction of the architecture while allowing further exploration in either all or part of the lower levels of abstraction for the architecture
- Provides a single modeling framework that can be used by design teams focused on different levels of abstraction of the system while allowing an automated approach to combining the efforts of each team
- Enhancing communication between the design teams through the use of a consistent modeling framework among all the teams

The modeling framework also ensures that all interesting designs from the element level are transferred up for investigation of the mission-level. By ensuring that the element-level models include property variables to account for not just the mission-level property variables, but also, for the relevant logical constraints for the element mapping decision, the framework provides a set of 'dominant' element designs based both on the property variables and the connections between the element and other decisions. This ensures that while a certain design may not be 'dominant' based on the property variables alone, it is seen as dominant (or at least interesting for investigation at the mission-level) based on both the mission-level property variables and the logical constraints in the model.

The multi-level decision-based system architecture framework not only preserves the validity of the decision-support viewing tools associated with the framework, it also may provide some benefits in terms of computational efficiency. Table 75 shows the statistics, it terms of architectures produced, for the multi-level modeling process compared to a single mixed-level representation that investigates the identical trade space including the system level and subsystem level decisions. Not only does the mixed-level representation reduce the validity of the decision-support viewing tools since the decisions are no longer on the same level of abstraction, it has significantly more unconstrained architectures that must be searched for feasible architectures and significant more feasible architectures that must be compared to locate the Pareto architectures.

Table 75. Statistics for the multi-level models compared to a single mixed-level representation.

	Unconstrained Architectures	Feasible Architectures	Pareto Architectures
Mixed mission level	**160**	**60**	**3**
Multilevel - Total	**128**	**37**	**6**
Habitat Level	80	20	3
Mission Level	48	17	3

Since the most computational intensive step of the simulation is searching the feasible architectures for the Pareto architectures, the computational efficiency of multi-level modeling compared to mixed level modeling is dependent on the ratio of Pareto architectures to feasible architectures at the element level. If a large portion of the feasible architectures are non-dominated architectures at the element level (Level 2), it raises the number of feasible alternatives at the mission-level (Level 1) to the point that the combined number of feasible architectures for the two models exceeds that of a mixed level model, which leads to a more computationally intensive simulation. If the ratio of

Pareto architectures to feasible architectures at the element model is less than a certain value, the multi-level modeling process will be more computationally efficient. For a system with no logical constraints at the element or mission-level, this value is calculated by Equation 4.

Equation 4. Cut-off ratio for computational efficiency for multi-level modeling.

$$\frac{\prod_{d_i \in F_{SP}} |a_{d_i}| - 1}{\prod_{d_i \in F_{SP}} |a_{d_i}| \times \prod_{d_j \in F_{FM}} |a_{d_j}|}$$

$d_i \in F_{SP}$: for all system parameter setting decisions

$d_j \in F_{FM}$: for all form - function mapping decisions not related to the element in question

$|a_{d_i}|$: number of decision alternatives for decision i

5.5 Summary

This chapter has introduced the concept of multi-level decision-based modeling. The chapter outlined the three steps required for multi-level modeling and showed the application of the theory to the same example problem discussed in Chapter Two. Section 5.4 discussed how multi-level modeling ensures validity of the decision viewing tools and ensures that a comprehensive search of the trade space is completed as well as discussing when multi-level decision-based modeling is more computationally efficient than a mixed-level modeling framework. The next chapter will apply the theory of multi-level decision-based modeling to a further investigation of deep space mission architectures.

6 Deep Space Design Reference Missions: A Case Study in Multi-level Decision-Based System Architecture Modeling

6.1 Introduction

Chapter Four introduced the reader to the concept of system architecting for deep space design reference missions through the application of a single-level decision-based system architecture model to the design of a human NEO exploration mission. Figure 49 recalls the generic mission profile that was utilized for the four NEO mission types investigated. The model examined the trade space for the operations required from departure from LEO until Earth re-entry an as such the model investigated form to function decisions related to the three propulsion systems required as well as the in-space habitat and the re-entry vehicle.

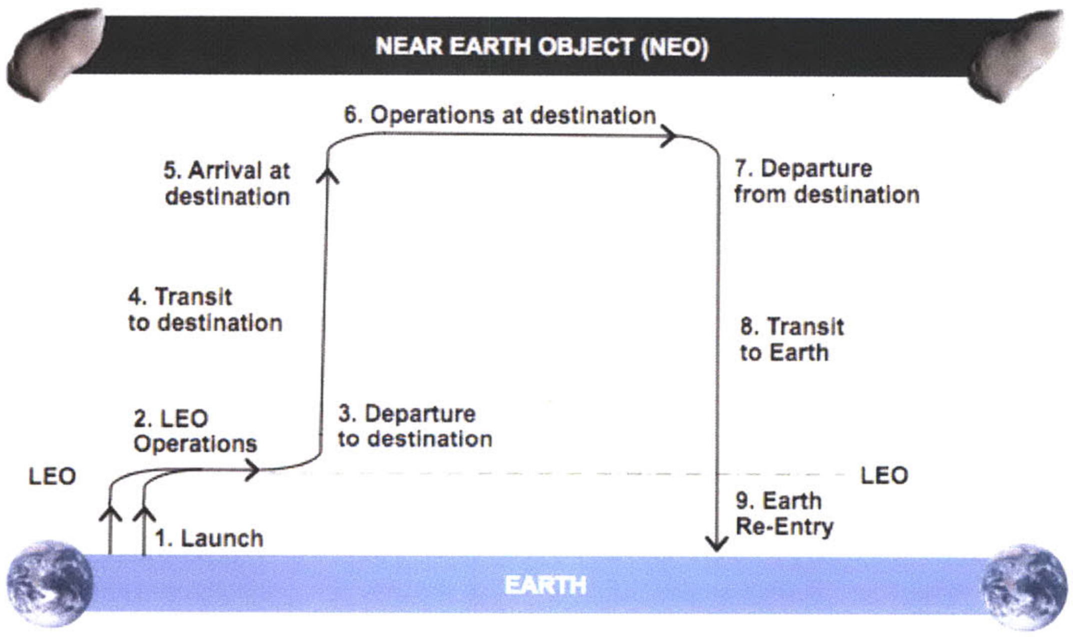

NEAR EARTH OBJECT (NEO)
6. Operations at destination
5. Arrival at destination
7. Departure from destination
4. Transit to destination
8. Transit to Earth
2. LEO Operations
3. Departure to destination
LEO
LEO
9. Earth Re-Entry
1. Launch
EARTH

Figure 49. Generic mission profile for all NEO missions.

Chapter Five introduced the concept of multi-level decision-based system architecture modeling and demonstrated the concept through the investigation of an example problem related to human spaceflight. Based on the importance of thoroughly exploring the trade space related to deep space missions, such as human NEO exploration missions, as set out in the FY2011 NASA budget proposal (NAS10), it would be beneficial to further investigate specific elements and their designs that are utilized by a NEO mission. In particular, the objective of this chapter is to further examine the in-space habitation model, and its trade space as it relates to NEO missions, through the application of the multi-level decision-based system architecture modeling framework.

This chapter follows the outline of Chapter Five as the multi-level model is developed. Section 6.2 briefly recaps the development of the initial mission-level model for NEO missions as laid out in Chapter Four. Section 6.3 outlines the development of an element-level model for the in-space habitat so that it can be used to inform the initial mission model. Section 6.4 links the multiple levels of models in order to gain further insight into the architectures and decisions. The chapter concludes with a summary presented in section 6.5.

6.2 Developing the Mission-Level Model

Chapter Four discussed the development of the initial mission-level model for the deep space design reference mission case study with a focus on the trade space related to human exploration of NEOs. The reader is directed to review Chapter Four for the background details related on developing the initial mission-level model. Based on the initial investigation of the mission-level, it has been decided to further examine the in-space habitation element.

In the initial mission-level model, the decision variable related to the function of providing in-space habitation was formulated with the alternatives as shown in Table 76. Each of the alternatives enumerated presented a generic habitat capable of supporting a given amount of crew for a given amount of days. It was assumed that a 180-day habitat could also support missions of longer durations. In addition, the habitation decision variable was connected to two other decisions through logical constraints as shown in Table 77.

Table 76. Habitation decision variable as presented in the initial mission-level model.

Decision	Alt A	Alt B	Alt C	Alt D	Alt E	Alt F
How is in-space habitation provided?	2crew 90day habitat	4crew 90day habitat	6crew 90day habitat	2crew 180day habitat	4crew 180day habitat	6crew 180day habitat

Table 77. Mission-level logical constraints connected to the habitation decision.

Constraint Name	Decision Variables Impacted	Logical Constraint Equations
Constraint 1	numCrew, hab	(numCrew==A&&(hab==A||hab==D))|| (numCrew==B&&(hab==B||hab==E))|| (numCrew==C&&(hab==C||hab==F))
Constraint 2	missionType, hab	missionType==A||missionType==B|| (missionType==D&&(hab==D||hab==E||hab==F))|| (missionType==C&&(hab==A||hab==B||hab==C))

With the initial model created, the next step is to develop the element-level model for the habitat in order to allow more informed decision alternatives at the mission level. The next section discusses the development and simulation of an element-level model for the in-space habitat.

6.3 Developing the Element-Level Model

Developing the element-level model follows the procedure for developing any single-level decision-based model for any given system. For this model, the in-space habitation element is the system of interest. In order to ensure valid connections through the multi-level framework, the system architect must continuously refer to the mission-level model to ensure that the choice of property variables and decision variables is accurate and comprehensive. The next four subsections outline the four steps required in representing the habitat element as a decision-based model. These subsections are followed with a fifth that focus on the simulation of the model in advance of using the information to inform the mission-level.

6.3.1 Choosing the property variables

The first task in representing the system is to choose the property variables. For a multi-level model, this task starts with recalling the property variables and functions at the mission-level as shown in Table 78.

Table 78. Property functions for the NEO mission-level model.

Property Variable	Property Functions
Number of crew-days at the NEO	Number of crew characteristic x duration at NEO characteristic
Number of crew-days in space	Number of crew characteristic x duration in space characteristic
Useful payload available [mt]	Useful payload available characteristic
Development of methane propulsion	Development of methane propulsion characteristic
Initial Mass in Low Earth Orbit (IMLEO) [mt]	habitat mass characteristic + re-entry mass characteristic + useful payload available characteristic + destination departure propulsion mass + destination arrival propulsion mass + earth departure propulsion mass
# of unique projects required	sum(project development characteristic)
Development of nuclear propulsion	Development of nuclear propulsion characteristic

The habitation element is directly connected to two of the seven property variables at the mission-level. These two property variables are: the calculation of IMLEO and the calculation of the number of unique projects to be developed. For the calculation of IMLEO, the characteristic of interest at the mission-level is the overall mass of the habitat element. Therefore, the overall mass of the element should be a property variable at the element-level to provide the information required for this mission-level characteristic. For the calculation of the number of unique projects, a property variable is included at the element-level to include whether or not a unique element has to be produced to provide in-space habitation.

As seen in the previous section, the habitation element is also linked to two other mission-level decisions (number of crew and mission type, through mission duration) and therefore, property variables must be included at the element-level to ensure that all the unique alternatives for the mission-level appear as dominant architectures at the element level. A property variable investigating the maximum number of crew and the maximum duration that any given habitation element can support is also included. Table 79 summarizes the four property variables included at the element level.

Table 79. Property Variables for the habitat element-level model.

Property Variable
System Mass [mt]
Maximum number of crew supported
Maximum duration supported [days]
Number of unique mission-level element projects required

6.3.2 Choosing the decision variables

As with any decision-based model, decisions related to setting system parameters and decisions related to mapping form to function need to be included in the model. The selection of system parameter decisions is informed with knowledge of what concepts are investigated at the mission-level. Based on this, two system parameter decisions that should be included at the element-level include:

- How many crewmembers can the habitat element support?
- How many days can the habitat element support the crew in space?

In terms of selecting the form to function mapping decisions, it is necessary to review the internal functions that an in-space habitat must provide the crew in order to support them. For the purpose of this case study, the internal functions are assumed to include:

- Providing a structure to support the necessary pressurized volume for supporting the given number of crew for the given number of days
- Providing power to support the other internal functions within the element
- Providing communication capability along with necessary avionics
- Providing the necessary environmental control and life support capabilities

Based on these four internal functions for the habitation element, there are four necessary form-to-function mapping decision variables that must be included:

- How is the structural support for the pressurized volume provided?
- How is power provided within the habitat?
- How is the capability for communications provided?

- How are the critical life support functions provided?

For each of the form to function mapping decisions, alternatives are selected from feasible subsystem designs that have been presented in the literature [Wer99, Hof09]. Table 80 shows the decision variables and their alternatives for the habitation element model.

Table 80. Decision variables for the habitation element model.

Decision Variable	Alt A	Alt B	Alt C	Alt D
How many crew must be supported?	2	4	6	
What duration must the habitat operate for?	89	156	175	304
How is structural support for the pressurized volume provided?	Rigid structural subsystem	Inflatable structural subsystem		
How is power provided within the habitat?	Solar arrays with Li-Ion batteries	Solar arrays with Regenerative fuel cells	RTG	
How is the capability for communications provided?	SD audio communications system	HD high-bandwidth communications system		
How are the critical life support functions provided?	LiOH canisters Stored Water Cryogenic stored oxygen Condensing heat exchanger	4-bed molecular sieve Stored water Cryogenic stored oxygen Condensing heat exchanger	LiOH canisters Multifiltration Cryogenic stored oxygen Condensing heat exchanger	4-bed molecular sieve Multifiltration Cryogenic stored oxygen Condensing heat exchanger

Once the decision variables and alternatives have been enumerated the next step is to formalize the connections between the decision variables and the property variables.

6.3.3 Developing the property functions

The third task required representing a system, as a decision-based model, is to develop the property functions for the four property variables. Table 81 shows the four initial property variables and their related functions for this model.

Table 81. Property functions for the habitation element-level model.

Property Variable	Property Function
Maximum number of crew supported	Maximum number of crew characteristic
Maximum duration supported [days]	Maximum duration characteristic
Number of unique mission-level element projects required	sum(unique project characteristic)
System Mass [mt]	sum(Subsystem Mass) + Logistics mass

The first two property functions are relatively simple compared to the last two listed in the table. The number of unique mission-level element projects is challenging to capture at the subsystem level since each single habitat, regardless of complexity, represents a unique mission-level element. The problem is that this will not allow any penalty to be captured for choosing more complex, yet less mass intensive subsystem alternatives. Therefore, each subsystem alternative was considered a fraction of a unique mission-level project. The combination of subsystems that creates the least complex habitat provides a value of 1.0. Habitats with more complex systems have higher unique project counts.

The system mass variable is calculated based on Equation 5. Table 82 and Table 83 show the relevant characteristics of each decision alternative.

Equation 5. System mass equation for the habitat element

$$\text{System Mass} = \left(\sum_{all\_subsystems} a + b \times numCrew + c \times numWhr + d \times numM3 + e \times numCrew \times duration \right) + (5 \times numCrew \times duration)$$

where

$$numWhr = numCrew \times 5kW \times duration$$
$$numM3 = numCrew \times 1.3 \times (6.67 \times \log(duration) - 7.76)$$

Table 82. Characteristics for the habitation element-level model (1 of 2).

Decision Variable	Alternatives	Max number of crew	Max duration	Number of unique projects	Enhanced Comm Capability
How many crew must be supported?	2	2			
	4	4			
	6	6			
What duration must the habitat operate for?	89		89		
	156		156		
	175		175		
	304		304		
How is structural support for the pressurized volume provided?	Inflatable			0.5	
	Rigid			0.25	
How is power provided within the habitat?	PV/batteries			0.25	
	PV/Fuel Cells			0.5	
	RTG			0.75	
How is the capability for communications provided?	SD			0.25	0
	HD			0.5	1
How are the critical life support functions provided?	Alt A			0.25	
	Alt B			0.5	
	Alt C			0.5	
	Alt D			0.75	

Table 83. Characteristics for the habitation element-level model (2 of 2).

Decision Variable	Alternatives	a [kg]	b [kg/per]	c [kg/Whr]	d [kg/m3]	e [kg/per/d]
How many crew must be supported?	2					
	4					
	6					
What duration must the habitat operate for?	89					
	156					
	175					
	304					
How is structural support for the pressurized volume provided?	Inflatable	0	0	0	10	0
	Rigid	0	0	0	20	0
How is power provided within the habitat?	PV/batteries	0	630	5	0	0
	PV/Fuel Cells	0	630	1.43	0	0
	RTG	0	2670	0	0	0
How is the capability for communications provided?	SD	250	0	0	0	0
	HD	600	0	0	0	0
How are the critical life support functions provided?	Alt A	0	10	0	0	17.01
	Alt B	0	40	0	0	15.26
	Alt C	0	20	0	0	2.59
	Alt D	0	50	0	0	0.84

There is one characteristic that is not connected to any of the four original property variables: enhanced communications capability. Based on the alternatives available for the communication subsystem and the four initial property variables, it can be seen that there are no connections to ensure that the advanced HD subsystem would appear in the pareto set and be transferred to the mission-level model. This is because there are no property variables to capture the benefit of having an HD communication system. Therefore, the system architect of the habitat element would recommend to the mission-level architect to include a new property variable: presence of enhanced communication capability that would capture the benefit and use the aforementioned characteristic.

6.3.4 Enumerating the logical constraints

The final task in representing is enumerating the logical constraints to ensure that all combinations of decisions that are enumerated represent a feasible architecture. This element-level model is interesting in that there are no logical constraints. Every combination of subsystem and system parameter is feasible. Since there are no logical constraints, the habitation element model can now be simulated and input into the mission-level model.

6.3.5 Simulating the element-level model

With no logical constraints, the simulation of the model produces the 576 unconstrained combinations of decisions as 576 feasible architectures. There are 144 different habitats sized for each mission type (i.e. unique duration). Because of the insight gained from the initial mission-level model as described in Chapter Four, the system architect decides to focus on a single mission-type in order to reduce the computational requirements of analyzing all mission types. Since the initial model showed that the 304-day mission type provided the highest benefit and lowest costs of any mission type for human NEO exploration, the following analysis focuses on the habitat and mission-level architectures that satisfy the requirements of the 304-day mission.

Based on the property variables chosen and the characteristics of the decision alternatives, there are 36 dominant architectures in terms of habitat designs for the 304-day mission.

6.4 Linking the Mission-Level and the Element Level Models

As described in the previous chapter, the element-level dominant architectures are entered into the mission-level model as feasible alternatives for the relevant form to function mapping decision, in this case the decision of which form provides the habitation functionality. The property variables from the element-level model are used as the characteristics for the decision alternatives for the habitats. Any relevant logical constraints that are connected to the decision of interest, in this case the habitation decision, are re-written as necessary using the new characteristics for the decision alternatives. Finally, any new or revised property variables introduced at the mission-level, such as the bookkeeping for enhanced communications capability, are input into the model. Once these chances have been made, the computer simulation is ready to be run and the new, informed results can be analyzed.

6.5 Results of the Informed Mission Model

The mission-level model, focusing only on the 304-day mission type and with 36 alternatives for the habitation elements, produced 524,880 unconstrained architectures and 9,720 feasible architectures. Based on the eight property variables, there were 729 architectures in the pareto set. Using the utility curves discussed in Chapter Four, it is possible to identify the architectures with the highest benefit score, lowest cost score, and highest system value as done previously. Table 84 shows the top ten architectures based on their value scores.

From Table 84, it can be seen that four habitat alternatives are seen in the top ten architectures. Table 85 shows the details of the four specific habitat architectures. Based on analyzing these habitat architects, the following insights can be gained:

- The use of inflatable materials for the habitat provides system-level value
- The use of RTG power systems for the habitat does not appear to provide system-level value
- The inclusion of HD communications provides system-level value
- The inclusion of advanced ECLSS does not appear to provide system-level value

Table 84. Top ten ranked mission architectures based on value (benefit/cost).

missionType	numCrew	propCommonality	usefulPayload	earthDept	hab	destArr	destDept	reEntry	nucPropDev	methPropDev	Benefit score	Cost score	Value score
304 day	6	full	4	LOX/ LCH4	26	LOX/ LCH4	LOX/ LCH4	6 crew	N	Y	0.88	0.14	6.09
304 day	6	full	2	LOX/ LCH4	26	LOX/ LCH4	LOX/ LCH4	6 crew	N	Y	0.84	0.14	5.82
304 day	6	full	4	LOX/ LCH4	30	LOX/ LCH4	LOX/ LCH4	6 crew	N	Y	0.88	0.16	5.66
304 day	6	full	2	LOX/ LCH4	30	LOX/ LCH4	LOX/ LCH4	6 crew	N	Y	0.84	0.16	5.40
304 day	6	full	10	LOX/ LCH4	26	LOX/ LCH4	LOX/ LCH4	6 crew	N	Y	1.00	0.19	5.29
304 day	4	full	4	LOX/ LCH4	14	LOX/ LCH4	LOX/ LCH4	4 crew	N	Y	0.75	0.14	5.17
304 day	4	full	4	LOX/ LCH4	14	LOX/ LCH4	LOX/ LCH4	6 crew	N	Y	0.75	0.14	5.17
304 day	6	full	4	LOX/ LCH4	25	LOX/ LCH4	LOX/ LCH4	6 crew	N	Y	0.68	0.13	5.10
304 day	6	full	8	LOX/ LCH4	26	LOX/ LCH4	LOX/ LCH4	6 crew	N	Y	0.96	0.19	5.08
304 day	6	full	10	LOX/ LCH4	30	LOX/ LCH4	LOX/ LCH4	6 crew	N	Y	1.00	0.20	5.00

Table 85. Top ranked habitat architectures based on overall mission value score.

Arch #	14	25	26	30
structure decision	inflatable	inflatable	inflatable	inflatable
power decision	PV/batt	PV/batt	PV/batt	PV/FC
comm decision	HD	SC	HD	HD
ECLSS decision	LiOH/StoredWater	LiOH/StoredWater	LiOH/StoredWater	LiOH/StoredWater
numCrew Supported	4	6	6	6
duration supported	304	304	304	304
enhanced comm	1	0	1	1
unique project count	1.25	1	1.25	1.5
element mass	63.5	94.6	94.9	62.4

6.6 Summary

This chapter discussed the application of multi-level decision-based system architecture modeling to the investigation of system architectures for human NEO exploration missions. The multi-level framework allowed further insight to be gained into the definition of the habitation element for NEO missions. Through the use of the multi-level framework, the decision-support tools had their validity maintained and the system architect was able to model the system in a process paralleling how real-world design teams work.

In terms of computational efficiency, Table 86 shows that the multi-level decision-based framework was able to enable a 25% reduction in the number of unconstrained architectures explored as well as a 75% reduction in the number of feasible architectures enumerated and evaluated as compared to a mixed-level model that produced the same set of non-dominated architectures for the human NEO missions.

Table 86. Computational statistics for the simulation of the multi-level NEO mission model.

	unconstrained architectures	feasible architectures	non-dominated architectures
mixed-level model	**699,840**	**38,880**	**729**
multi-level model	**525,024**	**9,864**	**765**
habitat model	144	144	36
mission model	524,880	9,720	729

7 Conclusion

7.1 Summary

The objective of this research was to provide a framework for formulating and investigating the decisions related to current human spaceflight projects, thereby allowing a more comprehensive enumeration and evaluation of the options for those specific projects. The framework was applied to specific human spaceflight projects to provide insight into their respective architectures.

The research evolved the concept of decision-based system architecture modeling from a starting point based on the Architecture Decision Graph framework developed by Simmons. The importance of applying the system architecture concept of levels of abstraction to decision-based modeling was discussed and a framework for single-level decision-based modeling was developed.

The concept of a single-level decision-based system architecture model was applied to two case studies of interest to the human spaceflight community: the design of a Lunar Surface System and the design of a human NEO mission. The results of these case studies can be seen in Chapters Three and Four.

The second part of this research focused on expanding the decision-based framework to investigate multiple-level of abstraction within a single system. Through the development of a multi-level decision-based system architecture modeling framework, both mission and element-level decisions can be investigated while ensuring the validity of the decision-support tools and enabling the modeling effort to mirror current real-world design practices. The key to multi-level modeling was shown to be the connection between lower-level property variables and higher-level characteristics and relevant logical constraints. By aligning these variables, it is possible to populate the decision alternatives for the system-level form to function mapping decisions with the dominant architectures from the element-level model.

The multi-level decision-based modeling framework was applied to further investigate the design space for NEO exploration missions. In particular, a more in-depth review of possible designs for the in-space habitation module was investigated. The results from this case study can be seen in Chapter Six.

The conclusions of this research are:

- Top-level qualitative system goals can be formally transformed into quantitative property variables for the model
- Decisions variables for a system are a combination of system attribute setting decisions and form to function mapping decisions
- Property functions are developed based on the concept of the characteristics of each decision alternative
- Logical constraints can be classified as one of three types based on their objectivity and reasoning
- Multi-Attribute Utility Theory aids in the viewing of interesting architectures and their property variables
- The Average Influence Score enables analysis if the influence of decisions in an architecture over the entire system
- Multiple-levels of single-level decision-based models can be linked hierarchically
- Linking lower-level model property variables to both the relevant higher-level model characteristics and logical constraints enables passing of information between models
- Multi-level modeling maintains the validity of the decision support views

7.2 Future Work

Based on the research related to this thesis and the generation of both the single-level and multi-level decision-based frameworks, the following areas for future work are recommended:

- Further investigation of the use of MAUT in the selection of property variables and the ability to quantify the higher-level objectives and goals through weighting to eventually gain quantitative values for evaluating architectures based on the top-level themes. This could be taken a step further through the incorporation of stakeholder network theory [Reb05] to enable a single value quantity in terms of the central organization to be quantified.

- Further investigation on the impact of set selection on the decision-support viewing tools would benefit this research. The ability to state that decision influence is more important when investigated for either the entire feasible set of architectures, the set of non-dominated architectures, or some sub-set there within, would be beneficial.
- The investigation of more computationally efficient methods for determining the non-dominated architectures from the feasible set would improve the speed of simulations. The search for non-dominated architectures was by far the most time consuming process during simulation and any improvement in this area would aid the research.
- An investigation into the links between element-level decisions and mission-level property variables in terms of providing more robust decision-based viewing tools has potential to provide a system architect with more beneficial information.
- The creation of a user-friendly automated program for representing decision-based system architecture models would allow easier interaction with stakeholders and rapid adaption of the models as required.

Appendix A: MatLab Code for Example Problem

```
clc
clear all
close all

[arch,prop,archAndProp]=missionSimulation;

constraintCount=[3,3,2,1];

paretoSorter=[0;0;0;0;-1;1;0];

nondom=paretosort(archAndProp,paretoSorter);

n=1;
for m=1:length(nondom)

    paretoSet(:,n)=archAndProp(:,nondom(n));
    n=n+1;
end

paretoArch=[paretoSet(1:(size(arch,1)-
1),:);paretoSet(size(paretoSet,1),:)];
paretoProp=paretoSet((size(arch,1)):size(paretoSet,1),:);

paretoPVS=pvs(paretoArch,paretoProp);
fullPVS=pvs(arch,prop);

individualAIS=zeros(size(paretoPVS,1),size(paretoPVS,2));
AIS=zeros(1,size(paretoPVS,2));

for m=1:size(paretoPVS,1)
    for n=1:size(paretoPVS,2)

individualAIS(m,n)=0.5*(constraintCount(n)/max(constraintCo
unt))+(0.5*paretoPVS(m,n)/max(paretoPVS(m,:)));
    end
end

for n=1:size(individualAIS,2)
    AIS(n)=mean(individualAIS(:,n));
end

xmax=30;
ymax=8;
paretoPlot(prop(2,:),prop(1,:),xmax,ymax,...
    'Number of Crew Days','System Mass [mt]','Number of
Crew Days vs System Mass');
```

```
xmax=max(constraintCount);
ymax=4;
xtick=0:1:xmax;
ytick=0:1:ymax;
hold on
dsvPlot(constraintCount,paretoPVS(1,:),...
   xmax,ymax,xtick,ytick,'System Mass')
% Add Text
text(constraintCount(1),paretoPVS(1,1),'  numCrew
','HorizontalAlignment','right')
text(constraintCount(2),paretoPVS(1,2),'  duration
','HorizontalAlignment','right')
text(constraintCount(3),paretoPVS(1,3),'  habitat
','HorizontalAlignment','right')
text(constraintCount(4),paretoPVS(1,4),'  power
','HorizontalAlignment','right')
hold off

xmax=max(constraintCount);
ymax=6;
xtick=0:1:xmax;
ytick=0:1:ymax;
hold on
dsvPlot(constraintCount,fullPVS(1,:),...
   xmax,ymax,xtick,ytick,'System Mass')
% Add Text
text(constraintCount(1),fullPVS(1,1),'  numCrew
','HorizontalAlignment','right')
text(constraintCount(2),fullPVS(1,2),'  duration
','HorizontalAlignment','right')
text(constraintCount(3),fullPVS(1,3),'  habitat
','HorizontalAlignment','right')
text(constraintCount(4),fullPVS(1,4),'  power
','HorizontalAlignment','right')
hold off

xmax=max(constraintCount);
ymax=xmax*ceil(max(paretoPVS(2,:))/xmax);
xtick=[0:1:xmax];
ytick=[0:ymax/xmax:ymax];
hold on
dsvPlot(constraintCount,paretoPVS(2,:),...
     xmax,ymax,xtick,ytick,'Crew-Days')
% Add Text
text(constraintCount(1),paretoPVS(2,1),'  numCrew
','HorizontalAlignment','right')
text(constraintCount(2),paretoPVS(2,2),'  duration
','HorizontalAlignment','right')
text(constraintCount(3),paretoPVS(2,3),'  habitat
','HorizontalAlignment','right')
text(constraintCount(4),paretoPVS(2,4),'  power
','HorizontalAlignment','right')
hold off
```

```
function [arch,prop,archAndProp] = missionSimulation();

% Input the list of decisions and their alternatives
% (ordered based on ADGsort2)

d1=[1,2]; %  #crew (1 or 2)
d2=[1,2]; %  duration (7 or 14)
d3=[1,2,3]; % habitat (1=ConceptA, 2=ConceptB, 3=ConceptC)
d4=[1,2]; % power (1=7dayBattery, 2=14daybattery)


% Input the characteristics for the decision alternatives

% Column 1 = mass [mt] (0 entry means n/a)
% Column 2 = number of crew (0 entry means n/a)
% Column 3 = duration of mission [days] (0 entry means n/a)
% Rows indicate the characteristic values for each
alternative

d1char = [0,0
          1,2
          0,0];
d2char = [0,0
          0,0
          7,14];
d3char = [1.3,2.6,2.9
          0,0,0
          0,0,0];
d4char = [2,4
          0,0
          0,0];


% The following code enumerates all feasible architectures
for the system
% Constraints are inserted after the 'for' loop for the
last variable of
% interest
m=1; %Architecture counter

for X1=1:length(d1)
for X2=1:length(d2)
    if ((X2==1)||((X1==2)&&(X2==2))) %Constraint 4
for X3=1:length(d3)
    if ((X1==1)||((X1==2)&&((X3==2)||(X3==3)))) %Constraint
2
    if ((X1==1&&X2==1)||(X1==1&&X2==2&&(X3==2||X3==3))||...
        (X1==2&&X2==1&&(X3==2||X3==3))||...
        (X1==2&&X2==2&&X3==3)) % Constraint 3
for X4=1:length(d4)
    if ((X2==1)||(X2==2&&X4==2)) % Constraint 1

        arch(:,m)=[X1;X2;X3;X4;m];
        m=m+1;

    end
end
    end
    end
end
    end
end
end


for n=1:size(arch,2)
    mass=d3char(1,arch(3,n))+d4char(1,arch(4,n));
    crewdays=d1char(2,arch(1,n))*d2char(3,arch(2,n));
    prop(:,n)=[mass;crewdays;n];
end

archAndProp=[arch(1:(size(arch,1)-1),:);prop];

end
```

```
function [PVS] = pvs(arch,prop)

PVS=zeros(size(prop,1)-1,size(arch,1)-1);
for D=1:(size(arch,1)-1)

    for P=1:(size(prop,1)-1)

%Create counter for each possible decision alternatives (up
to 6).
a1=1;
a2=1;
a3=1;
a4=1;
a5=1;
a6=1;
D_A1=[];
D_A2=[];
D_A3=[];
D_A4=[];
D_A5=[];
D_A6=[];

len=length(unique(arch(D,1:(size(arch,2)-1))));

% For each decision alternative,
% create a vector of values for the given property
variable.

for i=1:size(arch,2)
    if arch(D,i)==1
        D_A1(a1)=prop(P,i);
        a1=a1+1;
    elseif arch(D,i)==2
        D_A2(a2)=prop(P,i);
        a2=a2+1;
    elseif arch(D,i)==3
        D_A3(a3)=prop(P,i);
        a3=a3+1;
    elseif arch(D,i)==4
        D_A4(a4)=prop(P,i);
        a4=a4+1;
    elseif arch(D,i)==5
        D_A5(a5)=prop(P,i);
        a5=a5+1;
    elseif arch(D,i)==6
        D_A6(a6)=prop(P,i);
        a6=a6+1;
    end
```

```
end

% Calculate mean of the metric for ALL feasible
architectures.
mean_P=mean(prop(:,P));

%Calculate the mean of the metric for all architectures
with a given
%alternative.
if isempty(D_A1)==0
    mean_A1=mean(D_A1);
else
    mean_A1=mean_P;
end

if isempty(D_A2)==0
    mean_A2=mean(D_A2);
else
    mean_A2=mean_P;
end

if isempty(D_A3)==0
    mean_A3=mean(D_A3);
else
    mean_A3=mean_P;
end

if isempty(D_A4)==0
    mean_A4=mean(D_A4);
else
    mean_A4=mean_P;
end

if isempty(D_A5)==0
    mean_A5=mean(D_A5);
else
    mean_A5=mean_P;
end

if isempty(D_A6)==0
    mean_A6=mean(D_A6);
else
    mean_A6=mean_P;
end

%Calculate the PVS value for the given decision variable.
```

```
PVS(P,D)=sum([abs(mean_P-mean_A1) abs(mean_P-mean_A2) ...
    abs(mean_P-mean_A3) abs(mean_P-mean_A4) abs(mean_P-
mean_A5)...
    abs(mean_P-mean_A6)])/len;

    end
end

end
```

Appendix B: MatLab Code for LSS Case Study

```
function [arch,prop,archAndProp] = LSSsimulation()

tic
fprintf('Starting enumeration.\n')

% Input the list of decisions and their alternatives

d_num_mission=[1,2,3];        % Decision:  How many crew
missions will
                              % occur for this phase of lunar
exploration
                              % Alternatives: 1 = 1 mission,
                              % 2 = 3 missions, 3 = 5
missions

d_num_mission_char=[...
1   3   5
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0];

d_duration=[1,2,3,4,5];       % Decision: What is the
duration of
                              % each of the crew missions
                              % Alternatives: 1 = 7 days, 2 =
14 days,...
                              % 3 = 28 days, 4 = 60 days, 5 =
180 days

d_duration_char=[...
0   0   0   0   0
0   0   0   1   1
7   14  28  60  180
0   0   0   0   0
```

```
0    0    0    0    0
0    0    0    0    0
0    0    0    0    0
0    0    0    0    0
0    0    0    0    0
0    0    0    0    0
0    0    0    0    0
0    0    0    0    0
0    1    1    1    1
0    0    0    0    0];

d_eva=[1,2,3];              % Decision: How many EVAs per
week will
                            % each crew member complete?
                            % Alternatives: 1 = 1 per week,
                            % 2 = 2 per week, 3 = 3 per
week

d_eva_char=[...
0    0    0
0    0    0
0    0    0
1    2    3
0    0    0
0    0    0
0    0    0
0    0    0
0    0    0
0    0    0
0    0    0
0    0    0
0    0    0
0    0    0];

d_infra_mob=[1,2,3];        % Decision: What element(s)
will
                            % provide infrastructure
mobility?
                            % Alternatives: 1 = no element,
                            % 2 = 1x heavy lift mobility
element,
                            % 3 = 2x heavy lift mobility
elements

d_infra_mob_char=[...
0    0    0
0    0    0
0    0    0
```

```
0   0   0
10  1000    1000
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   2300    4700
0   0   0
0   940 1100];

d_press_mob=[1,2,3];            % Decision: What element(s)
will
                                % provide pressurized crew
mobility?
                                % Alternatives: 1 = no element,
                                % 2 = 2x pressurized rovers,
                                % 3 = 4x pressurized rovers

d_press_mob_char=[...
0   0   0
0   0   0
0   0   0
0   0   0
10  100 100
0   26  52
0   0   0
0   0   0
0   1   1
0   0   0
0   0   0
0   8300    16600
0   0   0
0   2000    2500];

d_unpress_mob=[1,2,3];          % Decision: What element(s)
will provide
                                % unpressurized crew mobility?
                                % Alternatives: 1 = no element,
                                % 2 = 1x small unpressurized
rover,
                                % 3 = 2x small unpressurized
rovers

d_unpress_mob_char=[...
0   0   0
0   0   0
```

```
0   0   0
0   0   0
10  10  40
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   230 460
0   0   0
0   430 650];

d_mob_power=[1,2,3];           % Decision: What element(s)
will
                               % provide mobile power?
                               % Alternatives: 1 = no element,
                               % 2 = solar array, 3 = RPS

d_mob_power_char=[...
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   1300    550
0   0   0
0   630 1300];

d_hab=[1,2,3,4,5];             % Decision: What element(s)
will
                               % provide habitation?
                               % Alternatives: 1 = no element,
                               % 2 = 28-day single habitat,
                               % 3 = 60-day single habitat,
                               % 4 = 180-day assembled double
habitat,
                               % 5 = 180-day non-assembled
double habitat

d_hab_char=[...
0   0   0   0   0
```

```
0   0   0   0   0
0   0   0   0   0
0   0   0   0   0
0   0   0   0   0
0   55  110 165 165
0   0   0   0   0
0   0   1   1   1
0   0   0   0   0
0   0   0   0   0
0   0   0   0   0
0   7500    11000   13000   13000
0   0   0   0   0
0   1700    2000    2800    2800];

d_out_power=[1,2,3];          % Decision: What element(s)
will
                              % provide outpost power?
                              % Alternatives: 1 = no element,
                              % 2 = solar array, 3 = Fission
Surface Power

d_out_power_char=[...
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   0   0
0   1   1
0   15000   9000
0   0   0
0   2000    5100];

d_comm=[1,2];                 % Decision: What element(s)
will
                              % provide enhanced
communications?
                              % Alternatives: 1 = no element,
                              % 2 = 1x Communications
Terminal

d_comm_char=[...
0   0
0   0
```

```
0   0
0   0
0   0
0   0
0   1
0   0
0   0
0   0
0   0
0   600
0   0
0   580];

d_ISRU=[1,2];                   % Decision: What element(s)
will provide
                                % ISRU capability?
                                % Alternatives: 1 = no element,
                                % 2 = 2x Oxygen Production
Plants

d_ISRU_char=[...
0   0
0   0
0   0
0   0
0   0
0   0
0   0
0   0
0   0
0   1
0   0
0   900
0   0
0   840];

% The following code enumerates all feasible architectures
for the system
% Constraints are inserted after the 'for' loop for the
last variable of
% interest

unconstrained=length(d_num_mission)*length(d_duration)*leng
th(d_eva)*...
                length(d_infra_mob)*length(d_press_mob)*...

length(d_unpress_mob)*length(d_mob_power)*...
length(d_hab)*length(d_out_power)*length(d_comm)*...
                   length(d_ISRU);
arch=zeros(12,unconstrained+1);
m=1; %Architecture counter

for X_num_mission=1:length(d_num_mission);
for X_duration=1:length(d_duration);
for X_eva=1:length(d_eva);
for X_infra_mob=1:length(d_infra_mob);
for X_press_mob=1:length(d_press_mob);
if ((X_press_mob==1&&X_duration==1)||...
    (X_press_mob==2&&X_duration==2)||...
    (X_duration~=1&&X_duration~=2)) % Constraint 4
for X_unpress_mob=1:length(d_unpress_mob);
if ((X_duration==1&&X_unpress_mob==2)||(X_duration~=1))
%Constraint 6
for X_mob_power=1:length(d_mob_power);
if ((X_mob_power==1&&X_press_mob==1)||...
    (X_mob_power~=1&&X_press_mob~=1)) % Constraint 1
for X_hab=1:length(d_hab);
if
(((X_duration==1)&&(X_hab==1))||((X_duration==2)&&(X_hab==1
))||...

((X_duration==3)&&(X_hab==2))||((X_duration==4)&&(X_hab==3)
)||...
    ((X_duration==5)&&(X_hab==4||X_hab==5))) % Constraint 2
if (((X_hab==4)&&(X_infra_mob~=1))||(X_hab~=4)) %Constraint
3
for X_out_power=1:length(d_out_power);
if (((X_out_power==1)&&(X_hab==1))||...
    ((X_out_power~=1)&&(X_hab~=1))) % Constraint 5
for X_comm=1:length(d_comm);
    if (((X_duration==1)&&(X_comm==1))||...
           (X_duration~=1))
for X_ISRU=1:length(d_ISRU);
        if (((X_duration==1)&&(X_ISRU==1))||...
           (X_duration~=1))

arch(:,m) =
[X_num_mission;X_duration;X_eva;X_infra_mob;X_press_mob;...

X_unpress_mob;X_mob_power;X_hab;X_out_power;X_comm;...
             X_ISRU;m];
m=m+1;

end
end
```

```
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end


arch=arch(:,1:m-1);

fprintf('Finished enumeration.\n')
fprintf('There are %d feasible architectures ',m-1)
fprintf('out of %d unconstrained
architectures.\n',unconstrained);

fprintf('Starting evaulation.\n')
prop=zeros(18,size(arch,2));
for n=1:size(arch,2)
    numAstronauts = d_num_mission_char(1,arch(1,n))*4;
    num60dayMissions = d_duration_char(2,arch(2,n))*...
                       d_num_mission_char(1,arch(1,n));
    totalCrewTime = d_num_mission_char(1,arch(1,n))*...
                       d_duration_char(3,arch(2,n))*4*24;
    evaCrewTime =
floor((d_num_mission_char(1,arch(1,n))*...
                       d_duration_char(3,arch(2,n))/7*4)*...
                       d_eva_char(4,arch(3,n))*8);
    ivaCrewTime = floor(d_num_mission_char(1,arch(1,n))*...
                       d_duration_char(3,arch(2,n))*4*24-...
                       (d_num_mission_char(1,arch(1,n))*...
                       d_duration_char(3,arch(2,n))/7*4)*...
                       d_eva_char(4,arch(3,n))*8);
    explorationRange =
max([d_infra_mob_char(5,arch(4,n)),...

d_press_mob_char(5,arch(5,n)),...
d_unpress_mob_char(5,arch(6,n))]);
    pressVolume =
d_press_mob_char(6,arch(5,n))+d_hab_char(6,arch(8,n));
    communications = d_comm_char(7,arch(10,n));
    ECLSSexp = d_duration_char(3,arch(2,n))*...

d_num_mission_char(1,arch(1,n))*d_hab_char(8,arch(8,n));
    roverOpsExp = d_duration_char(3,arch(2,n))*...

d_num_mission_char(1,arch(1,n))*d_press_mob_char(9,arch(5,n
));
    ISRUexp = d_duration_char(3,arch(2,n))*...

d_num_mission_char(1,arch(1,n))*d_ISRU_char(10,arch(11,n));
    powerExp = d_duration_char(3,arch(2,n))*...
        d_num_mission_char(1,arch(1,n))*...
        d_out_power_char(11,arch(9,n));
    dustMitigationExp =
floor((d_duration_char(3,arch(2,n))*...
        d_num_mission_char(1,arch(1,n)))/7*...
        d_eva_char(4,arch(3,n)));
    mass =  d_infra_mob_char(12,arch(4,n))+...
            d_press_mob_char(12,arch(5,n))+...
            d_unpress_mob_char(12,arch(6,n))+...
            d_mob_power_char(12,arch(7,n))+...
            d_hab_char(12,arch(8,n))+...
            d_out_power_char(12,arch(9,n))+...
            d_comm_char(12,arch(10,n))+...
            d_ISRU_char(12,arch(11,n))+...
            d_num_mission_char(1,arch(1,n))*...
            d_duration_char(3,arch(2,n))*4*10;
    numCargoFlights = ceil((mass-
d_num_mission_char(1,arch(1,n))*1000)/14500);
    numCrewedFlights = d_num_mission_char(1,arch(1,n));
    utilizationMass =
numCargoFlights*14500+numCrewedFlights*1000-mass;
    lifeCycleCost =
numCargoFlights*1700+numCrewedFlights*2500+...
            d_infra_mob_char(14,arch(4,n))+...
            d_press_mob_char(14,arch(5,n))+...
            d_unpress_mob_char(14,arch(6,n))+...
            d_mob_power_char(14,arch(7,n))+...
            d_hab_char(14,arch(8,n))+...
            d_out_power_char(14,arch(9,n))+...
            d_comm_char(14,arch(10,n))+...
            d_ISRU_char(14,arch(11,n));
```

```
prop(:,n)=[numAstronauts;num60dayMissions;totalCrewTime;...

evaCrewTime;ivaCrewTime;explorationRange;pressVolume;...

communications;ECLSSexp;roverOpsExp;ISRUexp;powerExp;...

dustMitigationExp;numCargoFlights;numCrewedFlights;...
        utilizationMass;lifeCycleCost;n];
end

fprintf('Finished evaulation.\n')
archAndProp=[arch(1:(size(arch,1)-1),:);prop];
toc
end
```

```
clc
clear all
close all

[arch,prop,archAndProp]=LSSsimulation;

constraintCount=[0,5,0,1,2,1,1,3,1,1,1];

tic

fprintf('Finding utility scores.\n')

util=utility(prop);

archAndUtil=[arch(1:11,:);util];

fprintf('Finished finding utility scores.\n')

toc

tic

fprintf('Finding nondominated solutions.\n')

% paretoSorter=[0;0;0;0;0;0;0;0;0;0;0;...
%               1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;-1;0];

paretoSorter=[0;0;0;0;0;0;0;0;0;0;0;...
              -1;1;1;0];

nondom=paretoSort(archAndUtil,paretoSorter);

fprintf('Finished finding %d nondominated
solutions.\n',length(nondom))

toc

tic

fprintf('Making nondominated sets.\n')

paretoSet=zeros(size(archAndUtil,1),length(nondom));

n=1;
for m=1:length(nondom)

    paretoSet(:,n)=archAndUtil(:,nondom(n));
    n=n+1;
end

paretoArch=[paretoSet(1:(size(arch,1)-
1),:);paretoSet(size(paretoSet,1),:)];

paretoProp=paretoSet((size(arch,1)):size(paretoSet,1),:);

fprintf('Finished making nondominated sets.\n')

toc

tic

fprintf('Calculating PVS scores.\n')

paretoPVS=pvs(paretoArch,paretoProp);

fullPVS=pvs(arch,prop);

fprintf('Finished calculating PVS scores.\n')

toc

ymax=40000;
xmax=1;
paretoPlot(util(2,:),util(1,:),xmax,ymax,'Architecture
Utility',...
    'Life Cycle Cost [M$ FY04]','Utility vs Life Cycle
Cost')

bestUtilArchs=sortrows(util',2);
bestUtilArchs=bestUtilArchs';

n=bestUtilArchs(4,size(util,2));
bestUtilArch=archAndProp(:,n);

bestValueArchs=sortrows(util',3);
bestValueArchs=bestValueArchs';

n=bestValueArchs(4,size(util,3));
bestValueArch=archAndProp(:,n);

tic

fprintf('Calculating AIS score.\n')

individualAIS=zeros(size(paretoPVS,1)-1,size(paretoPVS,2));
```

```
benefitAIS=zeros(1,size(paretoPVS,2));
costAIS=zeros(1,size(paretoPVS,2));
AIS=zeros(1,size(paretoPVS,2));

for m=1:size(paretoPVS,1)
    for n=1:size(paretoPVS,2)

individualAIS(m,n)=0.5*(constraintCount(n)/max(constraintCo
unt))+(0.5*(paretoPVS(m,n))/(max(paretoPVS(m,:))));
    end
end

for n=1:size(individualAIS,2)
    benefitAIS(n)=mean(individualAIS(2,n));
    costAIS(n)=individualAIS(1,n);

AIS(n)=mean([mean(individualAIS(2,n)),individualAIS(1,n)]);
end

individualPVS=zeros(size(paretoPVS,1),size(paretoPVS,2));
meanPVS=zeros(1,size(paretoPVS,2));
benefitPVS=zeros(1,size(paretoPVS,2));
costPVS=zeros(1,size(paretoPVS,2));

for m=1:size(paretoPVS,1)
    for n=1:size(paretoPVS,2)

individualPVS(m,n)=(paretoPVS(m,n))/(max(paretoPVS(m,:)));
    end
end

for n=1:size(individualPVS,2)
    benefitPVS(n)=mean(individualPVS(2,n));
    costPVS(n)=individualPVS(1,n);

meanPVS(n)=mean([mean(individualPVS(2,n)),individualPVS(1,n
)]);
end

fprintf('Finished calculating AIS scores.\n')

toc

hold on
dsvPlot(constraintCount,paretoPVS(17,:),'Life Cycle Cost')
% Add Text
text(constraintCount(1),paretoPVS(17,1),'  numMissions
','HorizontalAlignment','right')
```

```
text(constraintCount(2),paretoPVS(17,2),'  duration
','HorizontalAlignment','right')
text(constraintCount(3),paretoPVS(17,3),'  numEVA
','HorizontalAlignment','right')
text(constraintCount(4),paretoPVS(17,4),'  infraMobility
','HorizontalAlignment','right')
text(constraintCount(5),paretoPVS(17,5),'  pressMobility
','HorizontalAlignment','right')
text(constraintCount(6),paretoPVS(17,6),'  unpressMobility
','HorizontalAlignment','right')
text(constraintCount(7),paretoPVS(17,7),'  mobilityPower
','HorizontalAlignment','right')
text(constraintCount(8),paretoPVS(17,8),'  habitat
','HorizontalAlignment','right')
text(constraintCount(9),paretoPVS(17,9),'  outpostPower
','HorizontalAlignment','right')
text(constraintCount(10),paretoPVS(17,10),'  comm
','HorizontalAlignment','right')
text(constraintCount(11),paretoPVS(17,11),'  ISRU
','HorizontalAlignment','right')
hold off
hold on
dsvPlot(constraintCount,meanPVS(1,:),'Average of All
Property Variables ')
% Add Text
text(constraintCount(1),meanPVS(1,1),'  numMissions
','HorizontalAlignment','right')
text(constraintCount(2),meanPVS(1,2),'  duration
','HorizontalAlignment','right')
text(constraintCount(3),meanPVS(1,3),'  numEVA
','HorizontalAlignment','right')
text(constraintCount(4),meanPVS(1,4),'  infraMobility
','HorizontalAlignment','right')
text(constraintCount(5),meanPVS(1,5),'  pressMobility
','HorizontalAlignment','right')
text(constraintCount(6),meanPVS(1,6),'  unpressMobility
','HorizontalAlignment','right')
text(constraintCount(7),meanPVS(1,7),'  mobilityPower
','HorizontalAlignment','right')
text(constraintCount(8),meanPVS(1,8),'  habitat
','HorizontalAlignment','right')
text(constraintCount(9),meanPVS(1,9),'  outpostPower
','HorizontalAlignment','right')
text(constraintCount(10),meanPVS(1,10),'  comm
','HorizontalAlignment','right')
text(constraintCount(11),meanPVS(1,11),'  ISRU
','HorizontalAlignment','right')
hold off
```

```
function [totalUtility] = utility(properties)

totalUtility=zeros(3,length(properties));

for m=1:length(properties)

% Number of Astronauts
if properties(1,m)==20
    util_1=1;
elseif properties(1,m)==12
    util_1=0.66;
else util_1=0.33;
end


% Number of 60+ day missions
if properties(2,m)==5
    util_2=1;
elseif properties(2,m)==3
    util_2=0.66;
elseif properties(2,m)==1
    util_2=0.33;
else util_2=0;
end

% Total Crew Time
if properties(3,m)>80000
    util_3=1;
elseif properties(3,m)<0
    util_3=0;
else util_3=(properties(3,m))/80000;
end

% EVA Crew Time
if properties(4,m)>10000
    util_4=1;
elseif properties(4,m)<0
    util_4=0;
else util_4=(properties(4,m))/10000;
end

% IVA Crew Time
if properties(5,m)>60000
    util_5=1;
elseif properties(5,m)<0
    util_5=0;
else util_5=(properties(5,m))/60000;
end
```

```
% Mobility Range Enabled
if properties(6,m)==1000
    util_6=1;
elseif properties(6,m)==100
    util_6=0.75;
elseif properties(6,m)==40
    util_6=0.5;
elseif properties(6,m)==10
    util_6=0.25;
end

% Pressurized Volume
if properties(7,m)>200
    util_7=1;
elseif properties(7,m)<20
    util_7=0;
else util_7=(properties(7,m)-20)/180;
end

% Comm Bandwidth
if properties(8,m)==1
    util_8=1;
else util_8=0;
end

%ECLSS Ops
if properties(9,m)>800
    util_9=1;
elseif properties(9,m)<200
    util_9=0;
else util_9=(properties(9,m)-200)/600;
end

%Days Rover Ops

if properties(10,m)>800
    util_10=1;
elseif properties(10,m)<200
    util_10=0;
else util_10=(properties(10,m)-200)/600;
end

%ISRU Ops

if properties(11,m)>800
    util_11=1;
elseif properties(11,m)<200
```

```
    util_11=0;
else util_11=(properties(11,m)-200)/600;
end


% Power Experience

if properties(12,m)>800
    util_12=1;
elseif properties(12,m)<200
    util_12=0;
else util_12=(properties(12,m)-200)/600;
end

% Dust Mitigation

if properties(13,m)>300
    util_13=1;
elseif properties(13,m)<150
    util_13=0;
else util_13=(properties(13,m)-150)/150;
end

% number of Cargo Flights
if properties(14,m)>5
    util_14=1;
elseif properties(14,m)<2
    util_14=1;
else util_14=(properties(14,m)-2)/3;
end

% Number of crewed Landings
if properties(15,m)==5
    util_15=1;
elseif properties(15,m)==3
    util_15=0.66;
else util_15=0.33;
end

% Utilization Mass
if properties(16,m)>10000
    util_16=1;
elseif properties(16,m)<2000
    util_16=0;
else util_16=(properties(16,m)-2000)/8000;
end
```

```
util_total=(util_1+util_2+util_3+util_4+util_5+util_6+util_
7+util_8+...

util_9+util_10+util_11+util_12+util_13+util_14+util_15+...
            util_16)/16;


totalUtility(1,m)=properties(17,m);
totalUtility(2,m)=util_total;
totalUtility(3,m)=util_total/properties(17,m);
totalUtility(4,m)=properties(18,m);
end
end
```

Appendix C: MatLab Code for Single-Level Deep Space Case Study

```
clc
clear all
close all

tic

[arch,prop,archAndProp]=NEOsimulation;

toc

tic

fprintf('Finding utility scores.\n')

util=utility(prop);

fprintf('Finished finding utility scores.\n')

toc

tic
fprintf('Plotting Pareto Front View.\n')

paretoPlot(util(1,:),util(2,:),1,1,'Benefit','Cost','Benefi
t vs Cost');

fprintf('Finished plotting Pareto Front View.\n')

toc

tic

fprintf('Finding nondominated solutions.\n')

paretoSorter=[0;0;0;0;0;0;0;0;0;0;0;1;1;1;1;-1;-1;-1;0];

nondom=paretoSort(archAndProp,paretoSorter);

fprintf('Finished finding %d nondominated
solutions.\n',length(nondom))

toc
```

```
tic

fprintf('Making nondominated sets.\n')

n=1;
paretoSet=zeros(size(archAndProp,1),length(nondom));
for m=1:length(nondom)

    paretoSet(:,n)=archAndProp(:,nondom(n));
    n=n+1;
end

paretoArch=[paretoSet(1:(size(arch,1)-
1),:);paretoSet(size(paretoSet,1),:)];
paretoProp=paretoSet((size(arch,1)):size(paretoSet,1),:);

fprintf('Finished making nondominated sets.\n')

toc

tic

fprintf('Calculating PVS scores.\n')

paretoPVS=pvs(paretoArch,paretoProp);
fullPVS=pvs(arch,prop);

fprintf('Finished calculating PVS scores.\n')

toc

tic

fprintf('Plotting DSVs.\n')

constraintCount=[1,1,2,0,3,2,3,3,0,1,1];

hold on
dsvPlot(constraintCount,paretoPVS(1,:),'Crew Days @
Destination')
% Add Text
text(constraintCount(1),paretoPVS(1,1),'  missionType
','HorizontalAlignment','left')
text(constraintCount(2),paretoPVS(1,2),'  numCrew
','HorizontalAlignment','left')
text(constraintCount(3),paretoPVS(1,3),'  propCommon
','HorizontalAlignment','right')
```

```
text(constraintCount(4),paretoPVS(1,4),'  usefulPayload
','HorizontalAlignment','left')
text(constraintCount(5),paretoPVS(1,5),'  earthDept
','HorizontalAlignment','right')
text(constraintCount(6),paretoPVS(1,6),'  hab
','HorizontalAlignment','right')
text(constraintCount(7),paretoPVS(1,7),'  destArr
','HorizontalAlignment','right')
text(constraintCount(8),paretoPVS(1,8),'  destDept
','HorizontalAlignment','right')
text(constraintCount(9),paretoPVS(1,9),'  reEntry
','HorizontalAlignment','left')
text(constraintCount(10),paretoPVS(1,10),'  nucPropDev
','HorizontalAlignment','left')
text(constraintCount(11),paretoPVS(1,11),'  methPropDev
','HorizontalAlignment','left')
hold off

hold on
dsvPlot(constraintCount,paretoPVS(2,:),'Crew-days in
space')
% Add Text
text(constraintCount(1),paretoPVS(2,1),'  missionType
','HorizontalAlignment','left')
text(constraintCount(2),paretoPVS(2,2),'  numCrew
','HorizontalAlignment','left')
text(constraintCount(3),paretoPVS(2,3),'  propCommon
','HorizontalAlignment','right')
text(constraintCount(4),paretoPVS(2,4),'  usefulPayload
','HorizontalAlignment','left')
text(constraintCount(5),paretoPVS(2,5),'  earthDept
','HorizontalAlignment','right')
text(constraintCount(6),paretoPVS(2,6),'  hab
','HorizontalAlignment','right')
text(constraintCount(7),paretoPVS(2,7),'  destArr
','HorizontalAlignment','right')
text(constraintCount(8),paretoPVS(2,8),'  destDept
','HorizontalAlignment','right')
text(constraintCount(9),paretoPVS(2,9),'  reEntry
','HorizontalAlignment','left')
text(constraintCount(10),paretoPVS(2,10),'  nucPropDev
','HorizontalAlignment','left')
text(constraintCount(11),paretoPVS(2,11),'  methPropDev
','HorizontalAlignment','left')
hold off

hold on
dsvPlot(constraintCount,paretoPVS(3,:),'Useful payload mass
[mt]')
% Add Text
text(constraintCount(1),paretoPVS(3,1),'  missionType
','HorizontalAlignment','left')
text(constraintCount(2),paretoPVS(3,2),'  numCrew
','HorizontalAlignment','left')
text(constraintCount(3),paretoPVS(3,3),'  propCommon
','HorizontalAlignment','right')
text(constraintCount(4),paretoPVS(3,4),'  usefulPayload
','HorizontalAlignment','left')
text(constraintCount(5),paretoPVS(3,5),'  earthDept
','HorizontalAlignment','right')
text(constraintCount(6),paretoPVS(3,6),'  hab
','HorizontalAlignment','right')
text(constraintCount(7),paretoPVS(3,7),'  destArr
','HorizontalAlignment','right')
text(constraintCount(8),paretoPVS(3,8),'  destDept
','HorizontalAlignment','right')
text(constraintCount(9),paretoPVS(3,9),'  reEntry
','HorizontalAlignment','left')
text(constraintCount(10),paretoPVS(3,10),'  nucPropDev
','HorizontalAlignment','left')
text(constraintCount(11),paretoPVS(3,11),'  methPropDev
','HorizontalAlignment','left')
hold off

hold on
dsvPlot(constraintCount,paretoPVS(4,:),'Methane propulsion
development')
% Add Text
text(constraintCount(1),paretoPVS(4,1),'  missionType
','HorizontalAlignment','left')
text(constraintCount(2),paretoPVS(4,2),'  numCrew
','HorizontalAlignment','left')
text(constraintCount(3),paretoPVS(4,3),'  propCommon
','HorizontalAlignment','right')
text(constraintCount(4),paretoPVS(4,4),'  usefulPayload
','HorizontalAlignment','left')
text(constraintCount(5),paretoPVS(4,5),'  earthDept
','HorizontalAlignment','right')
text(constraintCount(6),paretoPVS(4,6),'  hab
','HorizontalAlignment','right')
text(constraintCount(7),paretoPVS(4,7),'  destArr
','HorizontalAlignment','right')
text(constraintCount(8),paretoPVS(4,8),'  destDept
','HorizontalAlignment','right')
text(constraintCount(9),paretoPVS(4,9),'  reEntry
','HorizontalAlignment','left')
```

```
text(constraintCount(10),paretoPVS(4,10),'  nucPropDev
','HorizontalAlignment','left')
text(constraintCount(11),paretoPVS(4,11),'  methPropDev
','HorizontalAlignment','left')
hold off

hold on
dsvPlot(constraintCount,paretoPVS(5,:),'IMLEO')
% Add Text
text(constraintCount(1),paretoPVS(5,1),'  missionType
','HorizontalAlignment','left')
text(constraintCount(2),paretoPVS(5,2),'  numCrew
','HorizontalAlignment','left')
text(constraintCount(3),paretoPVS(5,3),'  propCommon
','HorizontalAlignment','right')
text(constraintCount(4),paretoPVS(5,4),'  usefulPayload
','HorizontalAlignment','left')
text(constraintCount(5),paretoPVS(5,5),'  earthDept
','HorizontalAlignment','right')
text(constraintCount(6),paretoPVS(5,6),'  hab
','HorizontalAlignment','right')
text(constraintCount(7),paretoPVS(5,7),'  destArr
','HorizontalAlignment','right')
text(constraintCount(8),paretoPVS(5,8),'  destDept
','HorizontalAlignment','right')
text(constraintCount(9),paretoPVS(5,9),'  reEntry
','HorizontalAlignment','left')
text(constraintCount(10),paretoPVS(5,10),'  nucPropDev
','HorizontalAlignment','left')
text(constraintCount(11),paretoPVS(5,11),'  methPropDev
','HorizontalAlignment','left')
hold off

hold on
dsvPlot(constraintCount,paretoPVS(6,:),'Number of unique
development projects')
% Add Text
text(constraintCount(1),paretoPVS(6,1),'  missionType
','HorizontalAlignment','left')
text(constraintCount(2),paretoPVS(6,2),'  numCrew
','HorizontalAlignment','left')
text(constraintCount(3),paretoPVS(6,3),'  propCommon
','HorizontalAlignment','right')
text(constraintCount(4),paretoPVS(6,4),'  usefulPayload
','HorizontalAlignment','left')
text(constraintCount(5),paretoPVS(6,5),'  earthDept
','HorizontalAlignment','right')
text(constraintCount(6),paretoPVS(6,6),'  hab
','HorizontalAlignment','right')
text(constraintCount(7),paretoPVS(6,7),'  destArr
','HorizontalAlignment','right')
text(constraintCount(8),paretoPVS(6,8),'  destDept
','HorizontalAlignment','right')
text(constraintCount(9),paretoPVS(6,9),'  reEntry
','HorizontalAlignment','left')
text(constraintCount(10),paretoPVS(6,10),'  nucPropDev
','HorizontalAlignment','left')
text(constraintCount(11),paretoPVS(6,11),'  methPropDev
','HorizontalAlignment','left')
hold off

hold on
dsvPlot(constraintCount,paretoPVS(7,:),'Development of
nuclear propulsion')
% Add Text
text(constraintCount(1),paretoPVS(7,1),'  missionType
','HorizontalAlignment','left')
text(constraintCount(2),paretoPVS(7,2),'  numCrew
','HorizontalAlignment','left')
text(constraintCount(3),paretoPVS(7,3),'  propCommon
','HorizontalAlignment','right')
text(constraintCount(4),paretoPVS(7,4),'  usefulPayload
','HorizontalAlignment','left')
text(constraintCount(5),paretoPVS(7,5),'  earthDept
','HorizontalAlignment','right')
text(constraintCount(6),paretoPVS(7,6),'  hab
','HorizontalAlignment','right')
text(constraintCount(7),paretoPVS(7,7),'  destArr
','HorizontalAlignment','right')
text(constraintCount(8),paretoPVS(7,8),'  destDept
','HorizontalAlignment','right')
text(constraintCount(9),paretoPVS(7,9),'  reEntry
','HorizontalAlignment','left')
text(constraintCount(10),paretoPVS(7,10),'  nucPropDev
','HorizontalAlignment','left')
text(constraintCount(11),paretoPVS(7,11),'  methPropDev
','HorizontalAlignment','left')
hold off


fprintf('Finished plotting DSVs.\n')

toc

fprintf('Calculating AIS score.\n')
```

```
individualAIS=zeros(size(paretoPVS,1)-1,size(paretoPVS,2));
AIS=zeros(1,size(paretoPVS,2));

for m=1:size(paretoPVS,1)
    for n=1:size(paretoPVS,2)

individualAIS(m,n)=0.5*(constraintCount(n)/max(constraintCo
unt))+(0.5*paretoPVS(m,n)/max(paretoPVS(:,n)));
    end
end

for n=1:size(individualAIS,2)
    AIS(n)=mean(individualAIS(1:7,n));
end

individualPVS=zeros(size(paretoPVS,1),size(paretoPVS,2));
meanPVS=zeros(1,size(paretoPVS,2));

for m=1:size(paretoPVS,1)
    for n=1:size(paretoPVS,2)

individualPVS(m,n)=paretoPVS(m,n)/max(paretoPVS(:,n));
    end
end

for n=1:size(individualPVS,2)
    meanPVS(n)=mean(individualPVS(1:7,n));
end
fprintf('Finished calculating AIS scores.\n')

toc
```

```
function [arch,prop,archAndProp] = NEOsimulation()

%-------------------------------------------------------------------
--------------%
% The following section establishes for the decision
variables,
% their alternatives, and the related characteristics.
%-------------------------------------------------------------------
--------------%

d_missionType=...                % Decision: What mission
opportunity is being targeted?
    [1,2,3,4];                          %
Alternatives:1=NEO1999AO10 (in 2025) 2=NEO2001GP2 (in 2019)

d_missionType_char=...           % Characteristics for
mission type decision alternatives are:
    [111,130,29,70                      % Row1:Outbound duration
[days]
     14,14,30,16                % Row2:Duration at destination
[days]
     31,160,30,89                % Row3:Inbound duration [days]
     156,304,89,175                 % Row4: Total mission
duration [days]
     3.29,1.54,3.75,3.37                % Row5:DeltaV 1[km/s]
Earth departure
     2.19,2.09,5.01,1.21                % Row6:DeltaV 2[km/s]
Destination arrival
     1.75,0.17,4.05,1.16];                 % Row7:DeltaV
3[km/s] Destination departure

d_numCrew=...                    % Decision: How many crew
members on the mission?
    [1,2,3];                    % Alternatives: 1=2crew,
2=4crew, 3=6crew

d_numCrew_char=...               % Characteristics for
number of crew decision alternatives are:
    [2,4,6];            % Row1:Number of crew

d_propCommonality=...            % Decision: What degree of
commonality is used for propellant?
    [1,2,3];                     % Alternatives: 1=none,
2=deep space only, 3=all common

d_propCommonality_char=...
    [3,2,1];                     % Number of prop Projects
```

```
d_usefulPayload=...                 % Decision: How much useful
payload do you want to include?
    [1,2,3,4,5];                   % Alternatives: 1=2 mt,
2=4mt, 3=6mt, 4=8mt, 5=10mt

d_usefulPayload_char=...
    [2,4,6,8,10
     0,0,1,1,1];

d_earthDept=...                     % Decision: How is the
Earth departure propulsion function provided?
    [1,2,3];                        % Alternatives: 1=LOX/LH2
stage, 2=LOX/LCH4 stage, 3=NTR stage

d_earthDept_char=...                     % Characteristics for
Earth departure propulsion decision alternatives are:
    [450    369     900             % Row1: Specific Impulse
[sec]
     0.15   0.15    0.78];          % Row2: Structural Mass
Fraction (Inert Mass/propellant Mass)

d_hab=...                           % Decision: How is the in-
space habitation function provided?
    [1,2,3,4,5,6];                      % Alternatives:
1=Inflatable with ISS ECLSS, 2=Inflatable with Advanced
ECLSS, 3=Rigid with ISS ECLSS, 3=Rigid with Advanced ECLSS

d_hab_char=...                      % Characteristics for
habitat decision alternatives are:
    [2,4,6,2,4,6
     90,90,90,180,180,180
     18,25,33,21,32,43];

d_destArr=...                       % Decision: How is the
destination arrival propulsion function provided?
    [1,2,3];                        % Alternatives: 1=LOX/LH2
stage, 2=LOX/LCH4 stage, 3=NTR stage

d_destArr_char=...                  % Characteristics for
destination arrival propulsion decision alternatives are:
    [450    369     900             % Row1: Specific Impulse
[sec]
     0.15   0.15    0.78];          % Row2: Structural Mass
Fraction (Inert Mass/propellant Mass)

d_destDept=...                      % Decision: How is the
destination departure propulsion function provided?
```

```
    [1,2,3];                       % Alternatives: 1=LOX/LH2
stage, 2=LOX/LCH4 stage, 3=NTR stage

d_destDept_char=...                % Characteristics for
destination departure propulsion decision alternatives are:
    [450    369     900            % Row1: Specific Impulse
[sec]
     0.15   0.15    0.78];         % Row2: Structural Mass
Fraction (Inert Mass/propellant Mass)

d_reEntry=...                      % Decision: How is the
Earth re-entry function provided?
    [1,2,3];                          % Alternatives: 1=2
person Orion 2=4 person Orion 3=6person Orion

d_reEntry_char=...                 % Characteristics for re-
entry decision alternatives are:
    [2  4   6
     12 14 16];                     % Row1: Mass

d_nucPropDev=[1,2];

d_nucPropDev_char=[1,0];

d_methPropDev=[1,2];

d_methPropDev_char=[1,0];
%-----------------------------------------------------------
--------------%
% This section enumerates all feasible architectures based
on constraints.

tic

fprintf('Starting enumeration. \n');

m=1;                                  % 'm' is a counter for
the number of feasible architectures.
unconstrained=length(d_missionType)*length(d_numCrew)*...

length(d_propCommonality)*length(d_usefulPayload)*length(d_
earthDept)*...

length(d_hab)*length(d_destArr)*length(d_destDept)*...

length(d_reEntry)*length(d_nucPropDev)*length(d_methPropDev
);
arch=zeros(12,unconstrained);
```

```
for X_missionType=1:length(d_missionType)
for X_numCrew=1:length(d_numCrew)
for X_propCommonality=1:length(d_propCommonality)
for X_usefulPayload=1:length(d_usefulPayload)
for X_earthDept=1:length(d_earthDept)
for X_hab=1:length(d_hab)
if ((X_numCrew==1&&(X_hab==1||X_hab==4))||...
        (X_numCrew==2&&(X_hab==2||X_hab==5))||...
        (X_numCrew==3&&(X_hab==3||X_hab==6)))
if
(((X_missionType==1||X_missionType==2||X_missionType==4)&&.
..
         (X_hab==4||X_hab==5||X_hab==6))||...
        ((X_missionType==3)&&...
         (X_hab==1||X_hab==2||X_hab==3)))
for X_destArr=1:length(d_destArr)

for X_destDept=1:length(d_destDept)
if
((X_propCommonality==1)&&(X_destArr~=X_destDept)&&(X_earthD
ept~=X_destArr)&&(X_earthDept~=X_destDept))||...

(X_propCommonality==2&&((X_destDept==X_destArr&&X_earthDept
~=X_destArr)||...

(X_earthDept==X_destDept&&X_destArr~=X_destDept)||(X_earthD
ept==X_destArr&&X_destArr~=X_destDept)))||...

(X_propCommonality==3&&X_destDept==X_earthDept&&X_destDept=
=X_destArr)

for X_reEntry=1:length(d_reEntry)
if ((X_numCrew==1)||...
        (X_numCrew==2&&(X_reEntry~=1))||...
        (X_numCrew==3&&(X_reEntry==3)))
for X_nucPropDev=1:length(d_nucPropDev)
if  (X_nucPropDev==1&&X_earthDept==3)||...
    (X_nucPropDev==1&&X_destArr==3)||...
    (X_nucPropDev==1&&X_destDept==3)||...

(X_nucPropDev==2&&X_earthDept~=3&&X_destArr~=3&&X_destDept~
=3)
for X_methPropDev=1:length(d_methPropDev)
if  (X_methPropDev==1&&X_earthDept==2)||...
    (X_methPropDev==1&&X_destArr==2)||...
    (X_methPropDev==1&&X_destDept==2)||...
```

```
(X_methPropDev==2&&X_earthDept~=2&&X_destArr~=2&&X_destDept
~=2)


arch(:,m)=[X_missionType;X_numCrew;X_propCommonality;X_usef
ulPayload;X_earthDept;...

X_hab;X_destArr;X_destDept;X_reEntry;X_nucPropDev;X_methPro
pDev;m];
m=m+1;

end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end


arch=arch(:,1:m-1);

fprintf('Finished enumeration.\n')

fprintf('There are %d feasible architectures out ',
size(arch,2))
fprintf('of %d unconstrained
architectures.\n',unconstrained);

%-----------------------------------------------------------
--------------%

% This section evaluates the metrics for each feasible
architecture.

tic

fprintf ('Evaulating architectures. \n')
```

```
prop=zeros(8,size(arch,2));

for m=1:(size(arch,2))

    crewdaysDest=...

d_missionType_char(2,arch(1,m))*d_numCrew_char(1,arch(2,m))
;
    crewdaysSpace=...

d_missionType_char(4,arch(1,m))*d_numCrew_char(1,arch(2,m))
;
    usefulPayload=...
        d_usefulPayload_char(1,arch(4,m));
    % Initial Mass in LEO (IMLEO)


payloadMass=d_hab_char(3,arch(6,m))+d_reEntry_char(2,arch(9
,m))+...
        d_usefulPayload_char(1,arch(4,m));

    % Dest dept propulsion mass
        GRdept=exp(-
1000*d_missionType_char(7,arch(1,m))/...
            (d_destDept_char(1,arch(8,m))*9.81));

        SMFdept=d_destDept_char(2,arch(8,m));

        destDeptMass =(payloadMass)*...
            (1/(GRdept*(1+SMFdept)-SMFdept))...
            -(payloadMass);

    % Dest arr propulsion mass

         GRarr=exp(-
1000*d_missionType_char(6,arch(1,m))/...
            (d_destArr_char(1,arch(7,m))*9.81));

        SMFarr=d_destArr_char(2,arch(7,m));

        destArrMass=(payloadMass+destDeptMass)*...
            (1/(GRarr*(1+SMFarr)-SMFarr))...
            -(payloadMass+destDeptMass);


    %   Earth depart propulsion mass
    GRear=exp(-1000*(d_missionType_char(5,arch(1,m)))/...
        (d_earthDept_char(1,arch(5,m))*9.81));

    SMFear=d_earthDept_char(2,arch(5,m));

    earthDeptMass=(payloadMass+destDeptMass+destArrMass)...
    *(1/(GRear*(1+SMFear)-SMFear))...
    -(payloadMass+destDeptMass+destArrMass);

    %   IMLEO mass

IMLEO=payloadMass+destDeptMass+destArrMass+earthDeptMass;

    % unique projects


uniqueProjects=1+1+d_usefulPayload_char(2,arch(4,m))+...
        d_propCommonality_char(1,arch(3,m));
    nucPropDev=d_nucPropDev_char(1,arch(10,m));
    methPropDev=d_methPropDev_char(1,arch(11,m));
prop(:,m)=[crewdaysDest;crewdaysSpace;usefulPayload;methPro
pDev;IMLEO;uniqueProjects;nucPropDev;m];

end

archAndProp=[arch(1:(size(arch,1)-1),:);prop];

fprintf('Done evaluation. \n');
```

Appendix D: MatLab Code for Multi-Level Deep Space Case Study

```
function [uniqueArch,uniqueProp,uniqueArchAndProp] =
NEOhabitatSimulation()

% Input the list of decisions and their alternatives
% (ordered based on ADGsort2)


d1=[1,2,3]; %  #crew (2,4,6)
d2=[1,2,3,4]; %  duration (89,156,175,304)
d3=[1,2]; % structure (rigid,inflatable)
d4=[1,2,3]; %power (PV/batt, PV/FC, RTG)
d5=[1,2]; % comm (SD, HD)
d6=[1,2,3,4]; % ECLSS
(LIOH/stored,4bed/stored,LIOH/multif,4bed/mulitf)


% Input the characteristics for the decision alternatives

d1char = [2,4,6
          0,0,0
          0,0,0
          0,0,0
          0,0,0
          0,0,0
          0,0,0
          0,0,0
          0,0,0];
d2char = [0,0,0,0
          89,156,175,304
          0,0,0,0
          0,0,0,0
          0,0,0,0
          0,0,0,0
          0,0,0,0
          0,0,0,0
          0,0,0,0];
d3char = [0,0
          0,0
          0.25,0.5
          0,0
          0,0
          0,0
```

```
            0,0
            20,10
            0,0];
d4char = [0,0,0
            0,0,0
            0.25,0.5,0.75
            0,0,0
            0,0,0
            630,630,2670
            5,1.43,0
            0,0,0
            0,0,0];
d5char = [0,0
            0,0
            0.25,0.5
            0,1
            250,600
            0,0
            0,0
            0,0
            0,0];
d6char = [0,0,0,0
            0,0,0,0
            0.25,0.5,0.5,0.75
            0,0,0,0
            0,0,0,0
            10,40,20,50
            0,0,0,0
            0,0,0,0
            17.01,15.26,2.59,0.84];

% The following code enumerates all feasible architectures
for the system
% Constraints are inserted after the 'for' loop for the
last variable of
% interest
m=1; %Architecture counter

for X1=1:length(d1)
for X2=1:length(d2)
    if X2==4
for X3=1:length(d3)
for X4=1:length(d4)
for X5=1:length(d5)
for X6=1:length(d6)

        arch(:,m)=[X1;X2;X3;X4;X5;X6;m];
        m=m+1;
```

```
end
end
end
end
end
end
end

for n=1:size(arch,2)
    maxCrew=d1char(1,arch(1,n));
    maxDuration=d2char(2,arch(2,n));
    enhancedComm=d5char(4,arch(5,n));

projectCount=d3char(3,arch(3,n))+d4char(3,arch(4,n))+d5char
(3,arch(5,n))+d6char(3,arch(6,n));

    structureMass=...
        d3char(5,arch(3,n))+...
        d3char(6,arch(3,n))*d1char(1,arch(1,n))+...

d3char(7,arch(3,n))*d1char(1,arch(1,n))*5*d2char(2,arch(2,n
))*1+...

d3char(8,arch(3,n))*d1char(1,arch(1,n))*1.3*(6.67*log(d2cha
r(2,arch(2,n)))-7.76)+...

d3char(9,arch(3,n))*d1char(1,arch(1,n))*d2char(2,arch(2,n))
;

    powerMass=...
        d4char(5,arch(4,n))+...
        d4char(6,arch(4,n))*d1char(1,arch(1,n))+...

d4char(7,arch(4,n))*d1char(1,arch(1,n))*5*d2char(2,arch(2,n
))*1+...

d4char(8,arch(4,n))*d1char(1,arch(1,n))*1.3*(6.67*log(d2cha
r(2,arch(2,n)))-7.76)+...

d4char(9,arch(4,n))*d1char(1,arch(1,n))*d2char(2,arch(2,n))
;

    commMass=...
        d5char(5,arch(5,n))+...
        d5char(6,arch(5,n))*d1char(1,arch(1,n))+...

d5char(7,arch(5,n))*d1char(1,arch(1,n))*5*d2char(2,arch(2,n
))*1+...

d5char(8,arch(5,n))*d1char(1,arch(1,n))*1.3*(6.67*log(d2cha
r(2,arch(2,n)))-7.76)+...

d5char(9,arch(5,n))*d1char(1,arch(1,n))*d2char(2,arch(2,n))
;

    ECLSSMass=...
        d6char(5,arch(6,n))+...
        d6char(6,arch(6,n))*d1char(1,arch(1,n))+...

d6char(7,arch(6,n))*d1char(1,arch(1,n))*5*d2char(2,arch(2,n
))*1+...

d6char(8,arch(6,n))*d1char(1,arch(1,n))*1.3*(6.67*log(d2cha
r(2,arch(2,n)))-7.76)+...

d6char(9,arch(6,n))*d1char(1,arch(1,n))*d2char(2,arch(2,n))
;

systemMass=d1char(1,arch(1,n))*d2char(2,arch(2,n))*5+struct
ureMass+powerMass+commMass+ECLSSMass;

prop(:,n)=[maxCrew;maxDuration;enhancedComm;projectCount;sy
stemMass/1000;n];
end

archAndProp=[arch(1:(size(arch,1)-1),:);prop];
uniqueArchAndProp=archAndProp(3:(size(archAndProp,1)),:);

fprintf('Finished enumeration.\n')

unconstrained=length(d1)*length(d3)*length(d4)*length(d5)*l
ength(d6);
fprintf('There are %d feasible architectures out ',
size(arch,2))
fprintf('of %d unconstrained
architectures.\n',unconstrained);

% for n=1:(size(archAndProp,2)-1)
%     for m=n+1:size(archAndProp,2)
%
%         if archAndProp(3:(size(archAndProp,1)-
1),n)==archAndProp(3:(size(archAndProp,1)-1),m)
%             archAndProp(:,n)=zeros;
%         end
%     end
% end
%
% m=1;
% for n=1:size(archAndProp,2)
%     if archAndProp(:,n)~=zeros
%
uniqueArchAndProp(:,m)=archAndProp(3:size(archAndProp,1),n)
;
%         m=m+1;
%     end
% end

uniqueArch=[uniqueArchAndProp(1:4,:);uniqueArchAndProp(size
(uniqueArchAndProp,1),:)];
uniqueProp=uniqueArchAndProp(5:size(uniqueArchAndProp,1),:)
;
end
```

```
function [arch,prop,archAndProp] = informedNEOsimulation()

[habArch,habProp,habArchAndProp]=NEOhabitatSimulation;

paretoSorter=[0;0;0;0;1;1;1;-1;-1;0];

nondom=paretoSort(habArchAndProp,paretoSorter);

n=1;
for m=1:length(nondom)

    paretoSet(:,n)=habArchAndProp(:,nondom(n));
    n=n+1;
end

paretoHabArch=[paretoSet(1:(size(habArch,1)-
1),:);paretoSet(size(paretoSet,1),:)];
paretoHabProp=paretoSet((size(habArch,1)):size(paretoSet,1)
,:);

%------------------------------------------------------------
--------------%
% The following section establishes for the decision
variables,
% their alternatives, and the related characteristics.
%------------------------------------------------------------
--------------%

d_missionType=...                % Decision: What mission
opportunity is being targeted?
    [1,2,3,4];                        %
Alternatives:1=NEO1999AO10 (in 2025) 2=NEO2001GP2 (in 2019)

d_missionType_char=...           % Characteristics for
mission type decision alternatives are:
    [111,130,29,70                   % Row1:Outbound duration
[days]
     14,14,30,16             % Row2:Duration at destination
[days]
     31,160,30,89             % Row3:Inbound duration [days]
     156,304,89,175             % Row4: Total mission
duration [days]
     3.29,1.54,3.75,3.37             % Row5:DeltaV 1[km/s]
Earth departure
     2.19,2.09,5.01,1.21             % Row6:DeltaV 2[km/s]
Destination arrival
     1.75,0.17,4.05,1.16];              % Row7:DeltaV
```

```
3[km/s] Destination departure

d_numCrew=...                    % Decision: How many crew
members on the mission?
    [1,2,3];             % Alternatives: 1=2crew,
2=4crew, 3=6crew

d_numCrew_char=...               % Characteristics for
number of crew decision alternatives are:
    [2,4,6];          % Row1:Number of crew

d_propCommonality=...            % Decision: What degree of
commonality is used for propellant?
    [1,2,3];                     % Alternatives: 1=none,
2=deep space only, 3=all common

d_propCommonality_char=...
    [3,2,1];                     % Number of prop Projects

d_usefulPayload=...              % Decision: How much useful
payload do you want to include?
    [1,2,3,4,5];                % Alternatives: 1=2 mt,
2=4mt, 3=6mt, 4=8mt, 5=10mt

d_usefulPayload_char=...
    [2,4,6,8,10
     0,0,1,1,1];

d_earthDept=...                  % Decision: How is the
Earth departure propulsion function provided?
    [1,2,3];                     % Alternatives: 1=LOX/LH2
stage, 2=LOX/LCH4 stage, 3=NTR stage

d_earthDept_char=...                  % Characteristics for
Earth departure propulsion decision alternatives are:
    [450    369    900         % Row1: Specific Impulse
[sec]
     0.15   0.15   0.78];      % Row2: Structural Mass
Fraction (Inert Mass/propellant Mass)

d_hab=...                        % Decision: How is the in-
space habitation function provided?
    1:1:size(paretoHabArch,2);                   %
Alternatives: 1=Inflatable with ISS ECLSS, 2=Inflatable
with Advanced ECLSS, 3=Rigid with ISS ECLSS, 3=Rigid with
Advanced ECLSS

d_hab_char=...                   % Characteristics for
```

```
habitat decision alternatives are:
    [paretoHabProp(1,:)
     paretoHabProp(2,:)
     paretoHabProp(3,:)
     paretoHabProp(4,:)
     paretoHabProp(5,:)];

d_destArr=...                    % Decision: How is the
destination arrival propulsion function provided?
    [1,2,3];                     % Alternatives: 1=LOX/LH2
stage, 2=LOX/LCH4 stage, 3=NTR stage

d_destArr_char=...               % Characteristics for
destination arrival propulsion decision alternatives are:
    [450    369     900          % Row1: Specific Impulse
[sec]
     0.15   0.15    0.78];       % Row2: Structural Mass
Fraction (Inert Mass/propellant Mass)

d_destDept=...                   % Decision: How is the
destination departure propulsion function provided?
    [1,2,3];                     % Alternatives: 1=LOX/LH2
stage, 2=LOX/LCH4 stage, 3=NTR stage

d_destDept_char=...              % Characteristics for
destination departure propulsion decision alternatives are:
    [450    369     900          % Row1: Specific Impulse
[sec]
     0.15   0.15    0.78];       % Row2: Structural Mass
Fraction (Inert Mass/propellant Mass)

d_reEntry=...                    % Decision: How is the
Earth re-entry function provided?
    [1,2,3];                       % Alternatives: 1=2
person Orion 2=4 person Orion 3=6person Orion

d_reEntry_char=...               % Characteristics for re-
entry decision alternatives are:
    [2  4   6
     12 14 16];                   % Row1: Mass

d_nucPropDev=[1,2];

d_nucPropDev_char=[1,0];

d_methPropDev=[1,2];

d_methPropDev_char=[1,0];
```

```
%-------------------------------------------------------------
--------------%
% This section enumerates all feasible architectures based
on constraints.

tic

fprintf('Starting enumeration. \n');

m=1;                              % 'm' is a counter for
the number of feasible architectures.
unconstrained=1*length(d_numCrew)*...

length(d_propCommonality)*length(d_usefulPayload)*length(d_
earthDept)*...

length(d_hab)*length(d_destArr)*length(d_destDept)*...

length(d_reEntry)*length(d_nucPropDev)*length(d_methPropDev
);
arch=zeros(12,unconstrained);

for X_missionType=1:length(d_missionType)
    if (X_missionType==2)
for X_numCrew=1:length(d_numCrew)
for X_propCommonality=1:length(d_propCommonality)
for X_usefulPayload=1:length(d_usefulPayload)
for X_earthDept=1:length(d_earthDept)
for X_hab=1:length(d_hab)
if (d_numCrew_char(1,X_numCrew)==d_hab_char(1,X_hab))
if
(d_missionType_char(4,X_missionType)==d_hab_char(2,X_hab))
for X_destArr=1:length(d_destArr)
for X_destDept=1:length(d_destDept)
if
((X_propCommonality==1)&&(X_destArr~=X_destDept)&&(X_earthD
ept~=X_destArr)&&(X_earthDept~=X_destDept))||...

(X_propCommonality==2&&((X_destDept==X_destArr&&X_earthDept
~=X_destArr)||...

(X_earthDept==X_destDept&&X_destArr~=X_destDept)||(X_earthD
ept==X_destArr&&X_destArr~=X_destDept)))||...

(X_propCommonality==3&&X_destDept==X_earthDept&&X_destDept=
=X_destArr)
for X_reEntry=1:length(d_reEntry)
if ((X_numCrew==1)||...
        (X_numCrew==2&&(X_reEntry~=1))||...
        (X_numCrew==3&&(X_reEntry==3)))
for X_nucPropDev=1:length(d_nucPropDev)
if  (X_nucPropDev==1&&X_earthDept==3)||...
    (X_nucPropDev==1&&X_destArr==3)||...
    (X_nucPropDev==1&&X_destDept==3)||...

(X_nucPropDev==2&&X_earthDept~=3&&X_destArr~=3&&X_destDept~
=3)
for X_methPropDev=1:length(d_methPropDev)
if  (X_methPropDev==1&&X_earthDept==2)||...
    (X_methPropDev==1&&X_destArr==2)||...
    (X_methPropDev==1&&X_destDept==2)||...

(X_methPropDev==2&&X_earthDept~=2&&X_destArr~=2&&X_destDept
~=2)


arch(:,m)=[X_missionType;X_numCrew;X_propCommonality;X_usef
ulPayload;X_earthDept;...

X_hab;X_destArr;X_destDept;X_reEntry;X_nucPropDev;X_methPro
pDev;m];
m=m+1;

end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end
end


arch=arch(:,1:m-1);

fprintf('Finished enumeration.\n')
fprintf('There are %d feasible architectures out ',
size(arch,2))
fprintf('of %d unconstrained
architectures.\n',unconstrained);

%------------------------------------------------------------
--------------%

% This section evaluates the metrics for each feasible
architecture.

tic

fprintf ('Evaulating architectures. \n')

prop=zeros(9,size(arch,2));

for m=1:(size(arch,2))

    crewdaysDest=...

d_missionType_char(2,arch(1,m))*d_numCrew_char(1,arch(2,m))
;
    crewdaysSpace=...

d_missionType_char(4,arch(1,m))*d_numCrew_char(1,arch(2,m))
;
    usefulPayload=...
        d_usefulPayload_char(1,arch(4,m));
    % Initial Mass in LEO (IMLEO)


payloadMass=d_hab_char(5,arch(6,m))+d_reEntry_char(2,arch(9
,m))+...
        d_usefulPayload_char(1,arch(4,m));

    % Dest dept propulsion mass
        GRdept=exp(-
1000*d_missionType_char(7,arch(1,m))/...
            (d_destDept_char(1,arch(8,m))*9.81));

        SMFdept=d_destDept_char(2,arch(8,m));

        destDeptMass =(payloadMass)*...
            (1/(GRdept*(1+SMFdept)-SMFdept))...
            -(payloadMass);
    % Dest arr propulsion mass

         GRarr=exp(-
1000*d_missionType_char(6,arch(1,m))/...
            (d_destArr_char(1,arch(7,m))*9.81));

        SMFarr=d_destArr_char(2,arch(7,m));

        destArrMass=(payloadMass+destDeptMass)*...
            (1/(GRarr*(1+SMFarr)-SMFarr))...
            -(payloadMass+destDeptMass);


    %   Earth depart propulsion mass
    GRear=exp(-1000*(d_missionType_char(5,arch(1,m)))/...
        (d_earthDept_char(1,arch(5,m))*9.81));

    SMFear=d_earthDept_char(2,arch(5,m));

    earthDeptMass=(payloadMass+destDeptMass+destArrMass)...
    *(1/(GRear*(1+SMFear)-SMFear))...
    -(payloadMass+destDeptMass+destArrMass);

    %   IMLEO mass

IMLEO=payloadMass+destDeptMass+destArrMass+earthDeptMass;

    % unique projects


uniqueProjects=1+d_hab_char(4,arch(6,m))+d_usefulPayload_ch
ar(2,arch(4,m))+...
        d_propCommonality_char(1,arch(3,m));
    enhancedComm=d_hab_char(3,arch(6,m));
    nucPropDev=d_nucPropDev_char(1,arch(10,m));
    methPropDev=d_methPropDev_char(1,arch(11,m));
prop(:,m)=[crewdaysDest;crewdaysSpace;usefulPayload;methPro
pDev;IMLEO;uniqueProjects;nucPropDev;enhancedComm;m];

end


archAndProp=[arch(1:(size(arch,1)-1),:);prop];

fprintf('Done evaluation. \n');
```

Appendix E: LSS Needs-Goals-Objectives

THEME	GOAL	DESCRIPTION
Exploration Preparation	Understand the effects of the space environment to enable human exploration	Conduct research to monitor the impact of the integrated lunar surface environment on astronauts and systems in order to understand consequences of long-duration missions and to test mitigation strategies.
	Demonstrate and test technologies and systems that could be applicable to future exploration	Deploy equipment and systems on the lunar surface that will provide experience, data, and information that will support the design and execution of future exploration missions or which may be directly usable on future missions. Technologies and systems may be deployed and tested as part of operational lunar systems or as demonstration projects. Testing should include not only physical emplacement of technologies and systems but also adequate timelines to provide operational experience.
	Demonstrate and test operations that could be applicable to future exploration	Test operational concepts for specific activities that could support future exploration missions. Refine potential operations on the lunar surface with the intent of reducing uncertainty and risk in future missions.
	Gain experience in living and operating off of the Earth	Develop the experience and confidence in living in an off-Earth planetary environment that will be required to enable future exploration missions. Includes gaining experience in basic living tasks, operation of equipment, and operating autonomously from mission control.

Scientific Knowledge	Understand the formation, evolution and current state of the Moon	The Moon has been and will continue to be the scientific foundation for knowledge of the early evolution of the terrestrial planets. Use remotely sensed, geophysical, and sample data to allow scientists to define investigations that test and refine models established for lunar origin and evolution.
	Use the Moon as a "witness plate" for solar system evolution	As the Moon has been tectonically quiet over the last 3.8 Gy, it contains a record of extra lunar processes that occurred early in the history of the solar system to the present day. Investigate the record of processes stretching back to the first 500 My after solar system formation.
	Use the Moon as a platform for astrophysical, heliophysical, and earth-observing studies	The Moon provides a unique stable platform for observations of the Earth, the Sun and the Universe.
	Use the unique lunar environment as a research tool	The Moon has a unique combination of environmental characteristics, establishing experimental boundary conditions, not collectively attainable on Earth, that may be valuable and necessary to the investigation of high priority scientific questions. The following examples of lunar environmental characteristics should be considered illustrative and not exhaustive. For example, one significant and unique environmental characteristic is the long duration, steady 1/6 g environment present on the surface of the Moon. Many physical and biological systems are known to be sensitive to both the magnitude, direction, and temporal ("g-jitter") characteristics of gravity.

Human Civilization	Understand the effects of the lunar environment on humans	Conduct research to monitor the impact of the integrated lunar surface environment on astronauts and systems in order to understand consequences of long-duration lunar missions and to test mitigation strategies.
	Test technologies and systems that could support future human lunar expansion	Deploy equipment and systems on the lunar surface that will provide experience, data, and information that will support future lunar activities. Technologies and systems may be deployed and tested as part of operational lunar systems or as demonstration projects. Testing should include not only physical emplacement of technologies and systems but also adequate timelines to provide operational experience.
	Demonstrate and test operations that will allow for an expanded human lunar presence	Test operational concepts for specific activities that could support long-term expansion of the duration and self-sufficiency of human lunar operations.
	Gain experience in living and operating off of the Earth	Develop experience and confidence in living in an off-Earth planetary environment that will be required to enable expanded human presence on the moon. Includes gaining experience in basic living tasks, operation of equipment, and operating autonomously from mission control.
	Emplace infrastructure that allows for reuse for future human lunar expansion	Deploy lunar surface system architectures that could be reused to support future human lunar activities.
	Characterize and quantify the resource potential of the moon for the purposes of enabling settlement	Establish what types of resources can be developed on the lunar surface to support future human missions.
	Survey lunar geography and conditions for the purposes of determining settlement locations	Map conditions at varying points on the lunar surface. Mapping could include but is not limited to lighting, thermal, geography, terrain, electromagnetism.

Economic Expansion	Characterize and quantify the resource potential of the Moon for the purposes of economic expansion	Determine opportunities for feasible commercial activities on the lunar surface by establishing which goods and supplies could be created on the Moon.
	Develop architectures that provide additional capabilities that encourage commercial lunar activities	Ensure that the infrastructure provided as part of the lunar exploration program provides capabilities of interest to commercial entities, beyond what would be needed for Lunar Surface Systems usage.
	Provide opportunities for integration of lunar transportation elements from commercial entities	Ensure that the transportation architecture provides opportunities for commercially developed transportation systems and elements.
	Provide opportunities for integration of lunar surface and orbital elements from commercial entities	Ensure that the lunar surface architecture, including orbital assets, provides opportunities for commercially developed infrastructure systems and elements.
	Provide opportunities for integration of Earth-based services from commercial entities	Ensure that the lunar exploration architecture provides opportunities for commercially provided services from Earth.
	Incorporate technologies that have potential dual-use applications on Earth	Incorporate technologies that have the potential to provide commercial benefits through applications of the technology on Earth.

Global Partnerships	Characterize and quantify the potential resources on the Moon to guide future international activities	Determine opportunities for the future activities of international partners by providing them with detailed environmental data of the lunar surface and lunar assets.
	Develop architectures that provide additional capabilities that encourage future international lunar activities	Ensure that the infrastructure provided as part of the lunar exploration program provides capabilities of interest to international partners, beyond what would be needed for Lunar Surface Systems usage.
	Provide opportunities for integration of lunar transportation elements from international partners	Ensure that the transportation architecture provides opportunities for international transportation systems and elements.
	Provide opportunities for integration of lunar surface and orbital elements from international partners	Ensure that the lunar surface architecture, including orbital assets, provides opportunities for international infrastructure systems and elements.
	Provide opportunities for integration of Earth-based services from international partners	Ensure that the lunar exploration architecture provides opportunities for internationally provided services from Earth.

Public Engagement	Engage in events that excite the public	Establish an exploration architecture that produces as many media-worthy events as possible.
	Support media coverage of exploration	Ensure quality media access to the exploration program by establishing required capabilities and providing necessary communications resources and crew time availability.
	Provide opportunities for the public to interact with the architecture	Allow the public to participate in the exploration program by establishing required capabilities and providing necessary resources.
	Inspire STEM educators and students in exploration	Ensure that students and educators have access to observing and participating in the lunar exploration program by enabling the delivery of educational payloads.
	Provide stable and rewarding employment for the workforce	Create and maintain stable job opportunities for Americans in rewarding high-tech fields through the development, production and operation of the varied infrastructure required for exploration.

If the reader is interested in the related objectives for the listed goals, please feel free to email the author at guest.arthur@gmail.com.

References

[ATG08] M. Armstrong, C. de Tenorio, E. Garcia, and D. Mavris. *Function Based Architecture Design Space Definition and Exploration.* 26th Congress of International Council of the Aeronautical Sciences. AIAA 2008-8928.

[Aug09] N R. Augustine et al. *Seeking a Human Spaceflight Program Worthy of a Great Nation.* Final Report. Review of U.S. Human Spaceflight Plans Committee. 2009.

[BHW81] R.H. Bonczek, C.W. Holsapple, and A.B. Whinston. *Foundations of Decision Support Systems.* Academic Press, New York, 1981.

[BL85] Ronald J. Brachman and Hector J. Levesque, editors. *Readings in Knowledge Representation.* Morgan Kaufmann Publishers, 1985.

[Bra96] R.D. Braun. *Collaborative Architecture for Large-Scale Distributed Design.* PhD Dissertation. Aeronautics and Astronautics Dept. Stanford University. Stanford CA, 1996.

[Bro96] N.F. Brown and J.R. Olds. *Evaluation of Multidisciplinary Optimization Techniques Applied to a Reusable Launch Vehicle.* Journal of Spacecraft and Rockets. Vol. 43, No. 6 Nov-Dec 2006.

[Bus04] George W. Bush. *A Renewed Spirit of Discovery: The President's Vision for U.S. Space Exploration.* The White House. Washington, DC. 2004.

[Cat06] Sandro N. Catanzaro. *Multi-stakeholder Quantitative Analysis of Sustainability for Value Delivery Systems.* Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, June 2006.

[Cra07] Edward F. Crawley. ESD.34 *Systems Architecting – lecture notes.* MIT Engineering Systems Division, IAP 2007. http://ocw.mit.edu/courses/engineering-systems-division/esd-34-system-architecture-january-iap-2007/

[CO95] Zvi Covaliu and Robert M. Oliver. *Representation and solution of decision problems using sequential decision diagrams.* Management Science, 41(12):1860–1881, 1995.

[Coo08] Cooke, D. et al, *Lunar Architecture Update.* Proceedings. 3rd Space Exploration Conference. February 2008.

[DAE05] C.L. Dym, A.M Agogino, O. Eris, D.D. Frey, and L.J. Leifer. *Engineering Design Thinking, Teaching, and Learning.* Journal of Engineering Education, 94(1); 103-120. 2005.

[Dec04] Rina Dechter. *AND/OR search spaces for graphical models.* Technical report, UCLA ICS, March 2004.

[Dor02] Dov Dori. *Object-Process Methodology – A Holistic Systems Paradigm.* Springer Verlag, New York 2002. ISBN 3-540-65471-2.

[Dra97] B.G. Drake. *Addendum to the Human Exploration of Mars: The Reference Mission of the NASA Mars Exploration Study Team.* NASA. June 1998

[Dra09] B.G. Drake. *Human Exploration of Mars: Design Reference Architecture 5.0.* NASA-SP-2009-556. July 2009

[DSM] Design structure matrix (DSM). Website: http://www.dsmweb.org.

[Dye92] J.S. Dyer et al. *Multiple Criteria Decision Making, Mutli-Attribute Utility Theory: The Next Ten Years.* Management Science. Vol. 38, No. 5, May 1992.

[Eck07] Peter Eckart. *The Lunar Base Handbook.* Space Technology Series. 2007.

[Epp94] S.D. Eppinger, D.E. Whitney, R.P. Smith, and D.A. Gebala. *A model-based method for organizing tasks in product development.* Research in Engineering Design, 6(1):1–13, 1994.

[Han05] Sven Ove Hansson. *Decision theory: a brief introduction.* Technical report, Royal Institute of Technology (KTH) – Department of Philosophy and the History of Technology, Stockholm, 2005.

[Hof97] S.J. Hoffman and D.J. Kaplan. *Human Exploration of Mars: The Reference Mission of the NASA Mars Exploration Study Team.* NASA. July 1997

[Hof07] Wilfried Hofstetter et al. *The Intermediate Outpost – An Alternate Concept for Human Luanr Exploration.* Massachusetts Institute of Technology. Cambridge MA. AIAA-2007-6274 AIAA Space 2007 Conference and Exposition, Long Beach, California, Sep 18-20, 2007.

[Hof09] Wilfried K. Hofstetter. *A Framework for the Architecting of Aerospace Systems Portfolios with Commonality.* Massachusetts Institute of Technology. PhD Thesis. April 2010.

[HM84] Ronald A. Howard and James E. Matheson. *Influence diagrams.* In Ronald A. Howard and James E. Matheson, editors, Readings on the Principles and Applications of Decision Analysis, pages 721–762. Strategic Decisions Group, Menlo Park, California, 1984.

[HM05] Ronald A. Howard and James E. Matheson. *Influence diagrams. Decision Analysis*, 2(3):127–143, September 2005.

[JSC09] NASA JSC. *Cost Estimating Web Site: Online Cost Models.* Website: http://cost.jsc.nasa.gov/models.htm. Accessed 2009.

[KDL05] K. Kask, R. Dechter, J. Larrosa, and A. Dechter. Unifying tree decompositions for reasoning in graphical models. Artificial Intelligence, 166(1-2):165–193, 2005.

[Koo05] Benjamin H.Y. Koo. *A Meta-language for Systems Architecting.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2005.

[Lin10] Maokai Lin. Multi-objective Constrained Optimization for Decision Making and Optimization for System Architectures. Masters thesis, Massachusetts Institute of Technology, Cambridge, MA, 2010.

[Mat07] Robert Eugeniu Mateescu. AND/OR Search Spaces for Graphical Models. PhD thesis, University of California, Irvine, June 2007.

[Men85] Wendell Mendall. *Lunar Bases and Space Activities of the 21st Century.* The Lunar and Planetary Institute. Houston, TX. 1985.

[MR02] Mark W. Maier and Eberhardt Rechtin. *The Art of Systems Architecting.* CRC Press, 2nd Edition, 2002.

[NAS88] NASA, *1988 Case Studies.* Final Report 1988.

[NAS89] NASA. *1989 Case Studies.* Final Report 1989.

[NAS93] NASA. *First Lunar Outpost.* Final Report. 1993.

[NAS96] NASA. *Early Lunar Resource Utilization.* Final Report 1996.

[NAS02] NASA. *Exploration Blueprint Databook.* Final Report. November 2002.

[NAS05] NASA. *Exploration Systems Architecture Study Team.* Final Report. November 2005.

[NAS07] NASA. *Lunar Architecture Team Study 1.* LAT1. January 2007.

[NAS09] NASA. *Constellation Architecture Team Study.* CxAT. January 09.

[PB95] G. Pahl and W. Beitz. *Engineering Design: A Systematic Approach.* Springer, 2nd edition, 1995

[Pow02] Daniel J. Power. *Decision Support Systems: Concepts and Resources for Managers.* Quorum Books, 2002.

[PW00] Panos Y. Papalambros and Douglass J. Wilde. *Principles of Optimal Design: Modeling and Computation.* Cambridge University Press, 2nd edition, 2000

[Rai68] Howard Raiffa. *Decision Analysis: Introductory Lectures on Choices Under Uncertainty.* Addison-Wesley, 1968.

[RE09] D. Rayside and H.C. Estler. *A Spreadsheet-like User Interface for Combinatorial Multi-Objective Optimization.* 2009 Conference of the Center for Advanced Studies on Collaborative Research. 2009.

[Reb05] Rebentisch et al, 2005: *Using Stakeholder Value Analysis to Build Exploration Sustainability* AIAA 2005-2553 1st Space Exploration Conference, Orlando, Florida. 30 January - 1 February 2005.

[REJ09] D. Rayside, H.C. Estler,and D. Jackson. *The Guided Improvement Algorithm for Exact, General-Purpose, Many-Objective Combinatorial Optimization.* MIT Computer Science and Artificial Intelligence Laboratory Technical Report. MIT-CSAIL-TR-2009-033. July 3, 2009.

[Rit06] T. Ritchey. *Problem structuring using computer-aided morphological analysis*. Journal of the Operational Research Society, 57:792–801, 2006.

[RN02] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition, 2002.

[SdW07] Matthew R. Silver and Olivier L. de Weck. *Time-Expanded Decision Networks: A framework for designing evolvable complex systems*. Systems Engineering, 10(2):167–188, 2007.

[Sim60] Herbert A. Simon. *The New Science of Management Decision*. Harper and Brothers, New York, 1960.

[Sim77] Herbert A. Simon. *The New Science of Management Decision*. Prentice-Hall, Englewood Cliffs, N.J., 3rd edition, 1977

[Sim08] Willard L. Simmons. *A Framework for Decision Support in Systems Architecting*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 2008.

[Sob98] J. Sobieszanski-Sobieski, J. Agte, and R. Sandusky. *Bi-Level Integrated System Synthesis*. AIAA Paper 1998-4916. Sept 1998.

[Sob03] J. Sobieszanski-Sobieski, T Altus, M. Phillips, and R Sandusky. *Bi-Level Integrated System Synthesis for Concurrent and Distributed Processing*. AIAA Journal. Vol. 41, No. 10, 2003, pp. 1996-2003.

[TAL05] Efraim Turban, Jay E. Aronson, and Ting-Peng Liang. *Decision Support Systems and Intelligent Systems*. Prentice Hall, Upper Saddle River, New Jersey, 7th edition, 2005.

[Wer99] James Richard Wertz. *Space Mission Analysis and Design*. Springer,3rd edition, 1999.

[Zwi66] Fritz Zwicky. *Discovery, Invention, Research through the morphological approach*. Macmillan, 1966.