1st Space Exploration Conference: Continuing the Voyage of Discovery
30 January - 1 February 2005, Orlando, Florida

AIAA 2005-2543

A Quantitative Methodology for Identifying Evolvable Space Systems

John A. Christian, III[*] and John R. Olds[†]
Georgia Institute of Technology, Atlanta, GA, 30332

With the growing emphasis on spiral development, a system's ability to evolve is becoming increasingly critical. This is especially true in systems designed for the exploration of space. While returning to the Moon is widely regarded as the next step in space exploration, our journey does not end there. Therefore, the technologies, vehicles, and systems created for near-term lunar missions should be selected and designed with the future in mind. Intelligently selecting evolvable systems requires a method for quantitatively measuring evolvability and a procedure for comparing these measurements. This paper provides a brief discussion of a quantitative methodology for evaluating space system evolvability and an in-depth application of this methodology to an example case study.

Nomenclature

e	=	event
s	=	state vector of original mission requirements
s'	=	state vector of evolved mission requirements
$S(X)$	=	possible state space of system
$S_L(X)$	=	lawful state space of system
$\mathbf{W}$	=	evolvability vector
w_i	=	difficulty rating for i-th state variable along most efficient path of evolvability

I. Introduction

RECENT events have turned the eyes of the United States back to the moon, this time as a stepping stone to missions beyond. In his speech at NASA Headquarters[1] in Washington, D.C. on January 14, 2004, President George W. Bush said that "returning to the moon is an important step for our space program. Establishing an extended human presence on the moon could vastly reduce the costs of further space exploration, making possible ever more ambitious missions." It is clear, therefore, that the systems developed to accomplish these near term lunar missions should lend themselves to adaptation in order to meet the requirements more ambitious missions. These systems must be evolvable. Indeed, the President's Commission on Implementation of United States Space Exploration Policy[2] recommended that NASA "design an exploration architecture that evolves iteratively, systematically through a series of so-called spiral developments."

It is, therefore, necessary to develop a reliable method for identifying evolvable space systems. This will allow evolvability to be included in the numerous figures of merit already used to select the optimal system out of a pool of numerous candidates. Recalling that one of the attributes of a good figure of merit is the ability to achieve a quantitative measurement[3], a quantitative methodology was chosen over a qualitative methodology.

II. Defining Evolvability

The first step in creating a metric to measure system evolvability is to precisely define the term "evolvability." Over the years, many definitions for evolvability have been formulated[4,5,6,7]. Based on these past definitions, the following definition of evolvability is proposed:

[*] Undergraduate Student, School of Aerospace Engineering, AIAA Student Member.
[†] Associate Professor, School or Aerospace Engineering, AIAA Associate Fellow.

Evolvability: The capacity of a system to adapt to changing requirements throughout its lifecycle without compromising the integrity of the system.

This simple definition, however, may be misleading- after all, evolvability may take many different forms and a given system may often accommodate change in a variety of ways. Evolvability, then, can be viewed as a composite of the many methods a system possesses for adapting to change. Understanding this functional breakdown, termed the "taxonomy of change" by Rowe, Leaney and Lowe[5], is fundamental to developing a quantitative measure of the larger property of evolvability. Evolvable systems may accommodate change primarily in one of two ways: statically or dynamically. Further inspection shows that a system can adapt to change in one of four ways: make no changes to the system (generality), reconfiguring existing system components (adaptability), increasing the size of existing system components (scalability), or adding new components (extensibility). These four methods of adapting to change will be referred to as the four classes of evolvability. Figure 1 graphically illustrates the functional breakdown of evolvability.

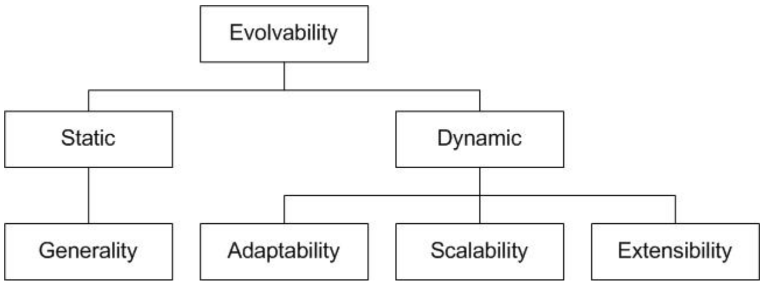

Figure 1. A functional breakdown of evolvability.

The four classes of evolvability, largely inspired by the work of Rowe, Leaney, and Lowe[5] and information available in the SMC Systems Engineering Primer and Handbook[3], are described in more detail in the definitions that follow. These classes of evolvability are defined by the ability of a system to accommodate a change in requirements through:

Generality: No change in the existing architectural design (i.e. the system already meets the evolved requirements).

Adaptability: Rearranging or duplicating existing system components within the current architecture.

Scalability: Increasing the size of architectural components to accommodate increased loads.

Extensibility: Adding new components that were not part of the original architecture.

III. Theoretical Support

Given a more precise definition of evolvability and a better understanding of the ways in which a system may evolve, we begin the process of developing a quantitative metric for evolvability. Any evolving system changes through exhibiting one or more of the four classes of evolvability, each of which may be quantified by representing independent mission attributes as state variables. In this new state space each mission is modeled as a discrete point, while each system occupies a space representing the suite of missions it is capable of performing. For example, when comparing multiple architectures capable of performing the same mission, a region should exist where the space occupied by these architectures overlap. The point representing the baseline mission should lie within this overlapping region if all the architectures are initially capable of performing the mission. In this approach, evolvability of a system is modeled through a transformation of the space occupied by that system. The manner in which this space is transformed is governed by the specific mode(s) of evolvability exhibited by the system.

This model of evolvability is largely based on the work of Mario Bunge[8]. Although Bunge's work is primarily philosophical, it is also mathematically rigorous, which has lead to research in its application to engineering and computer science[9,10,11]. An adaptation of Bunge's work to system evolvability is presented in the remainder of this section. This discussion is an abbreviated version of the theory to be presented in more detail in a follow-on paper.

Bunge asserts that the world is constructed of entities called 'things.' Furthermore, a thing may be either simple (consisting of only one thing) or composite (consisting of two or more things). A system may be defined as a composite thing that may not be broken down into non-interacting components. Many systems exhibit emergent properties (i.e. properties arising from the combination of things which cannot be directly traced to a property of one of the composite things), and evolution is an example of one of these emergent properties[12].

Because evolution is a type of change, it may be modeled as an event. An event, e, is defined as an ordered pair of states, with each state (beginning, s, and end, s') existing in the possible state space, S(*X*). The possible state space exists in *n* dimensions (where *n* is the number of state variables), and represents all possible mission combinations. An actual system, however, is not able to accomplish all possible missions, but rather a subset of these missions. This smaller space that defines the capabilities of a particular system is known as the lawful state space, $S_L(X)$. If the end state, s', is outside of the lawful state space, $S_L(X)$, then this change may not occur unless the lawful state space is modified to include s'. Therefore, an event is represented mathematically by the following expression:

$$e = \langle s, s' \rangle \quad \text{where } s, s' \in S(X) \tag{1}$$

The concept of modeling evolvability as an event implies that you must first have knowledge of both the initial and evolved states (a concept that will be crucial in developing a practical application of this theory). Figure 2 shows the possible modes of evolution that two states may define. The first is static evolvability, e_1: the evolved state lies within the lawful state space of the original system, $S_L(X)$. The second is dynamic evolvability, e_2: the evolved state does not lie within the lawful state space of the original system, but the lawful state space may be modified, $S_L(X')$, to include the evolved state. The third is a non-evolvable event, e_3: the evolved state does not lie within the lawful state space of the original system and the lawful state space may not be modified to include the evolved state; this event is not possible.

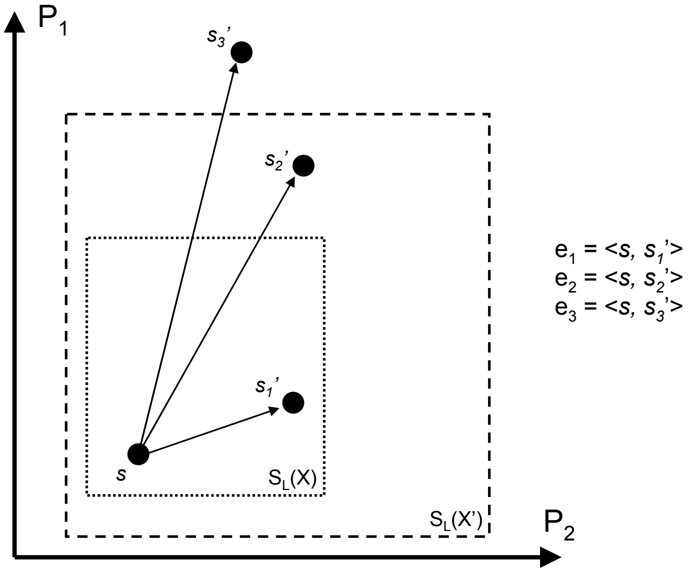

Figure 2. Graphical representation of system evolvability.

IV. Practical Application

Admittedly, this methodology is quite abstract and could be applied to the problem of evolution in many different ways. The approach presented in this section is but one of these many approaches. The selected approach will be presented in the context of a case study for the purposes of example and clarity. Here, it is important to point out that a more desirable evolvability score does not necessarily represent the best architecture to accomplish the mission at hand. Evolvability is only one of a number of important figures of merit. The architecture with the best evolvability score may also be too expensive or too risky.

A. Case Study Overview

This case study will examine an example scenario where the objective is to determine which of two different architectures used to execute the same lunar exploration mission is most evolvable to a substantially more ambitious lunar campaign. The first baseline architecture, representing a Libration Point Rendezvous (LPR) approach, is the OASIS architecture developed at NASA Langley Research Center. The original Langley OASIS architecture was augmented with a lunar lander element to provide lunar surface capability. The second baseline architecture, representing a Lunar Surface Rendezvous (LSR) approach, is the First Lunar Outpost. These two baseline architectures are compared and numerical scores for evolvability are computed for each. These scores represent each architecture's potential to evolve to the extended lunar mission; therefore, the architecture with the better score is more desirable from an evolutionary standpoint.

1. Libration Point Rendezvous Baseline Architecture

The Orbital Aggregation and Space Infrastructure Systems (OASIS)[13,14] architecture was developed by NASA's Langley Research Center (LaRC) in 2002. The purpose of OASIS is to create highly reusable infrastructure to service a wide array of potential missions. By staging crew, fuel, and supplies at the Earth-Moon L_1 (EML_1) point, the OASIS architecture provides a platform for lunar excursions and other Earth's neighborhood missions. The OASIS architecture begins with the assumption that the mission starts from the International Space Station (ISS) in Low Earth Orbit (LEO) and that a "lunar gateway" already exists at the EML_1. From here, four new vehicles are introduced: the Crew Transfer Vehicle (CTV), the Hybrid Propellant Module (HPM), the Chemical Transfer Module (CTM), and the Solar Electric Propulsion (SEP) Stage

Individually, any one of these vehicles is not very useful. Each one performs a different task required by the mission: the CTV carries a crew of four to the EML_1, the HPM carries the fuel, the CTM provides high-thrust chemical propulsion, and the SEP provides high-efficiency/low-thrust propulsion. For the purpose of this comparison, a lunar lander element was added to the original OASIS architecture to add lunar surface capability. The lander will carry a crew of four to and from the lunar surface for a duration of seven days. The lander can also carry 1,000 kg of cargo to the lunar surface. The architecture will nominally be able to perform one lunar excursion every six months. OASIS is also characterized by the reusability of its vehicles. Ideally, each vehicle should be reused for a number of missions before requiring replacement.

2. Lunar Surface Rendezvous Baseline Architecture

The First Lunar Outpost (FLO) architecture[15,16] was developed by NASA in 1992 as a proposed approach to return to the moon. FLO was designed to send the crew all the way to the lunar surface in a single launch (as opposed to the modular approach used by OASIS). The first manned mission begins by launching an unmanned, one-way cargo flight that contains a habitat and consumables. Once this element has safely landed, a crew of four launches from Earth and lands on the lunar surface near the predeployed habitat. Here, it is important to note that both the unmanned cargo mission and manned lander have a common descent stage. The crew will stay on the lunar surface for 45 days in the habitat before returning to the manned lander for the return trip to Earth. The manned lander is capable of delivering four crew members and 5,000 kg of cargo. For the purposes of this study, one crew will be sent to the lunar habitat every six months. Unlike OASIS, however, the vehicles in the FLO architecture are primarily expendable (with the exception of the hab). Each crew module and lander is only capable of supporting a single mission.

B. Selecting Evolvability State Variables for Space Systems

When selecting a set of state variables, one should always use the minimum number of state variables to fully define a mission and render it distinct from all other missions. The state variables selected should also be selected such that they independent of one another. Here, it is important to recall that the state variables represent characteristics of specific missions (not characteristics of the architectures). Also recall that the selected state variables from the basis for the possible state space in which all events will occur. A much more detailed discussion of state variable selection will be included in the follow-on paper discussed earlier.

Only thee state variables were chosen for this case study for the purposes of clarity and presentation. Clearly, more than three state variables are necessary to fully distinguish a particular mission. Unfortunately, more than three state variables may not be easily graphed, and a graphical representation of the state space helps illustrate the concept quite well. Therefore, for the purposes of this case study, the following three state variables were chosen:

1. Number of crew per mission
2. Duration on the lunar surface
3. Cargo mass to lunar surface per year

In the case of the lunar case study presented in this paper, the baseline values for the three state variables and the maximum expected value of these variables are shown in Table 1.

Table 1. State variable data for baseline architectures.

	Original Value		Max Expected
State Variable	LPR	LSR	Value
Number of crew per mission	4	4	8
Duration on the lunar surface (days)	7	45	180
Cargo to lunar surface per year (kg)	2,000	10,000	50,000

C. Calculating a System's Evolvability Score

Recall that a system may evolve either statically or dynamically through any of the four classes of evolvability. Because generality is basically the "do-nothing" alternative, it is the easiest class of evolvability to evaluate. A system will either be general enough to meet these new requirements or it will not, so static evolvability may be viewed in a binary fashion. The three classes of evolvability that describe dynamic evolvability, however, require a change in the capabilities (or lawful state space) of the system. Therefore, evaluating evolvability should begin by assessing how much work must be done to the stretch the original state space to engulf the evolved state.

1. Difficulty Matrices and the Difficulty Scale for Evolvability Analysis

The process of evaluating evolvability begins with assessing the difficulty of stretching a system's lawful state space along the direction of the *i*-th state variable. This difficulty, however, frequently depends on the characteristics of the evolved system, requiring knowledge of both the beginning and end states to assign a difficulty rating. Determining this metric may be done in a number of ways. The methodology chosen in this particular approach requires the construction of a difficulty matrix for each state variable for a given system (i.e. each matrix addresses a specific system and state variable combination). Each difficulty matrix will have four rows (each corresponding to one of the four classes of evolvability) and a number of columns equivalent to the number of values of the state variable for which results are desired. Each element within the matrix represents the difficulty of stretching the capability of the system to the value indicated by the column through the evolvability path indicated by the row.

The values that populate this matrix may be obtained quantitatively or by consensus. A quantitative approach is preferred for assigning difficulty ratings, but this method may not be reasonable due to time and budget constraints. Therefore, for the purposes of this paper and case study, the consensus approach was used. In this approach, a group of "experts" is asked to assign each of the difficulty ratings that populate the difficulty matrices. For this group to effectively complete this task, some set of guidelines must be established to ensure consistent and accurate scoring. To accomplish this, the Difficulty Scale for Evolvability Analysis (DSEA) was formulated and is presented in Table 2.

Table 2. The Difficulty Scale for Evolvability Analysis (DSEA).

Difficulty Rating	Explanation
1	Easy- *Requires little effort to evolve. Some new technology, but functionality of most system components remain unchanged. System complexity remains about the same.*
3	Moderate- *Requires a moderate amount of effort to evolve. Some new technology, the functionality of some system components change, and system complexity increases by a small amount.*
9	Difficult- *Requires a large amount of effort to evolve. Many new technologies, the functionality of many system components change, and there is a substantial increase in system complexity.*
27	Very Difficult- *Requires a very large amount of effort to evolve. Many new technologies are required, most system components change functionality, and system complexity is greatly increased.*
81	Infeasible- *The desired capabilities are prohibitively difficult, if not impossible, to accomplish..*

Despite this scoring scale, the group of experts may still find it difficult to assign reliable difficulty ratings. To aid such groups in developing these ratings, a brief discussion of the system attributes that lead to the different classes of evolvability is presented. To begin, Rowe, Leaney, and Lowe[5] provide some insight into the nature of the different classes of evolvability, shown in Fig. 3.

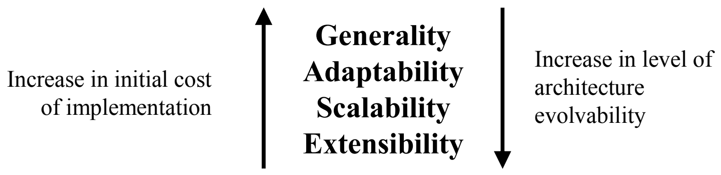
Increase in initial cost of implementation
Generality
Adaptability
Scalability
Extensibility
Increase in level of architecture evolvability

Figure 3. Relationship between different classes of evolvability[5].

In order to provide additional guidance, Table 3 includes comments and guidelines outlining a few things to keep in mind when assigning DSEA scores. In addition, a few useful definitions are listed below to help clarify the information presented in Table 3.

Modularity[17]: The degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.

Open Standards[3]: Parts, modules, objects, products and systems are based on vendor-independent, non-proprietary, publicly available, and widely accepted standards. Standards allow for a transparent environment where users can intermix hardware, software, and networks of different vintages from different vendors to meet different needs.

Open Architecture[18]: Hardware and software interfaces are defined such that additional resources can be added to the system with little or no adjustment.

Interoperability[3]: The ability of systems, units, or forces to provide and receive services from other systems, units, or forces and to use the services so interchanged to enable them to operate efficiently together.

Table 3. Characteristics of the three classes of dynamic evolvability.

Class of Evolvability	Common Design Characteristics
Adaptability	Modularity, open standards, standard interfaces Modularity is arguably more important in adaptability than in any other class of evolvability. This is because the essence of adaptability is rearranging existing elements (or modules) to achieve a new mission. If more modules are available, then more potential combinations for rearrangement may exist. Open standards and standard interfaces are also necessary in conjunction with modularity to make reconfiguration possible.
Scalability	Room for growth, modularity, open standards Scalability involves increasing the size of one or more of the components of the system. Modularity is useful in allowing the engineer to scale only the components necessary without altering the other components.
Extensibility	Modularity, open standards, open architecture, interoperability A system desiring to evolve effectively through extensibility should be built on the aspects of the system which are most likely to remain unchanged. These "islands of architectural stability"[6] enable a complex system to evolve much quicker and easier than what might otherwise be possible. Note that such islands of architectural stability suggest a modular approach, allowing the engineer to isolate core functionality in a set of stable, autonomous modules.

2. Construction of the Difficulty Matrices

Equipped with the tools to develop difficulty matrices and assign DSEA scores, we apply this procedure to the lunar case study. A group of 11 graduate students in the Space Systems Design Lab (SSDL) at the Georgia Institute of Technology were presented with this case study and asked to populate the required difficulty matrices. The difficulty matrices developed by this group for the LSR architecture are presented below in Table 4 through Table 6. The matrices for the LPR architecture we formulated in the same fashion but are not shown.

Table 4. Difficulty matrix for number of crew in the LSR (FLO) architecture.

	Number of Crew		
	4	6	8
Generality	1	81	81
Adaptability	1	1	1
Scalability	1	9	27
Extensibility	1	9	9

Table 5. Difficulty matrix for the duration on lunar surface in the LSR (FLO) architecture.

	Duration on Lunar Surface (days)		
	7	90	180
Generality	1	81	81
Adaptability	1	1	3
Scalability	1	3	9
Extensibility	1	3	9

Table 6. Difficulty matrix for cargo mass to lunar surface per year in the LSR (FLO) architecture.

	Cargo mass to lunar surface per year (kg)		
	1,000	25,000	50,000
Generality	1	81	81
Adaptability	1	3	9
Scalability	1	9	27
Extensibility	1	9	27

The consensus exercise performed by the SSDL graduate students was very successful, although a smaller group may have improved efficiency. Of particular interest was the group's tendency to always increase difficulty when moving from left to right (direction of increasing capability) in the difficulty matrices. While such a tendency is normal, scores may not always increase from left to right. There may be local minima or maxima in the difficulty scores and groups should be reminded of this at the beginning of the study.

3. Computing and Comparing Evolvability Scores

Given the opportunity, a system will evolve along the most efficient path (i.e. via the class of evolvability for which it is best suited). Indeed, architectures need not exhibit all classes of evolvability to effectively evolve. Excelling in only one class may be all that is necessary. To address this, the difficulty rating for the *i*-th state variable is given by the minimum EDSA score in the corresponding column. The minimum EDSA score for the *i*-th state variable is denoted by w_i. An example of this is shown in Fig. 4 using one of the difficulty matrices for the LSR architecture. Organizing these minimum EDSA scores into vector form yields the evolvability vector, **W**. A separate evolvability vector must be constructed for each mission and architecture combination.

$$\mathbf{W} = \begin{bmatrix} w_1 \\ w_2 \\ \mathrm{M} \\ w_n \end{bmatrix} \tag{2}$$

Note that in Fig. 4 the number of crew corresponds to the first state variable, meaning that the number of crew difficulty rating is w_1. Again, notice that the vectors shown in Fig. 4 represent the difficulty for only the LSR architecture, separate vectors must also be constructed for the LPR architecture.

	Number of Crew		
	4	6	8
Generality	1	81	81
Adaptability	1	1	1
Scalability	1	9	27
Extensibility	1	9	9

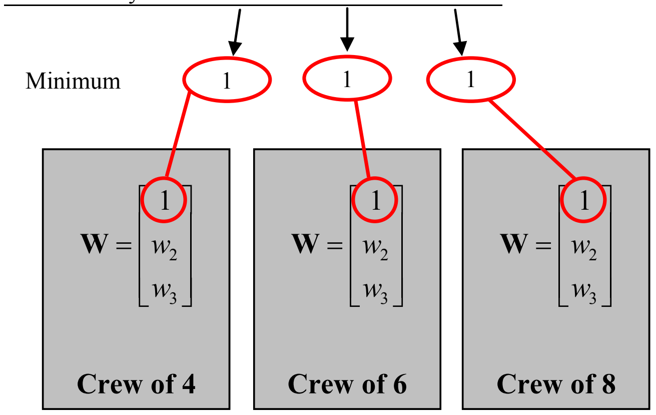

Figure 4. Populating the evolvability vector for three different LSR (FLO) scenarios.

Now, the evolvability score may be simply calculated using the following expression.

$$Evolvability\ \ Score = \left\|\mathbf{W}\right\| \tag{3}$$

When using this metric a lower score indicates a more desirable result. Applying this procedure to the LPR and LSR architectures in this lunar mission case study yields the results shown in Fig. 5 and Fig. 6. The low evolvability scores (green) represent areas to which the architecture can easily evolve, while high evolvability scores (red) indicate areas the architectures may not reach.

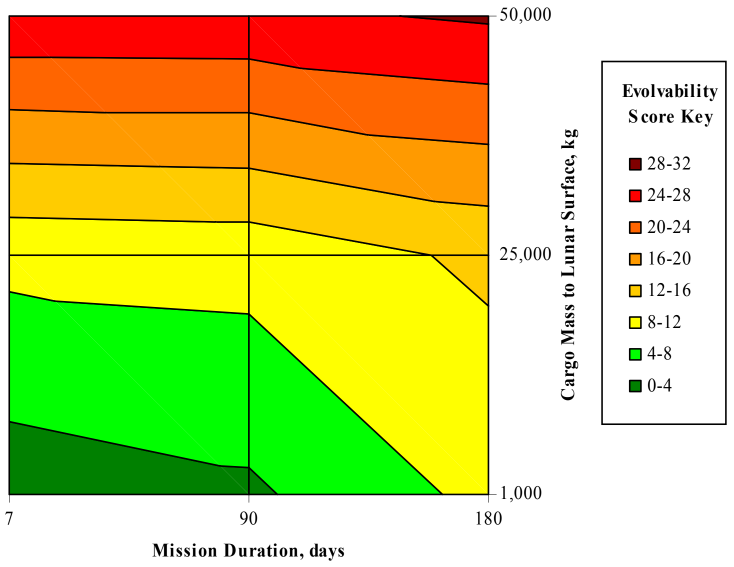

Figure 5. LPR architecture (OASIS) evolvability score contour plot for four crew members.

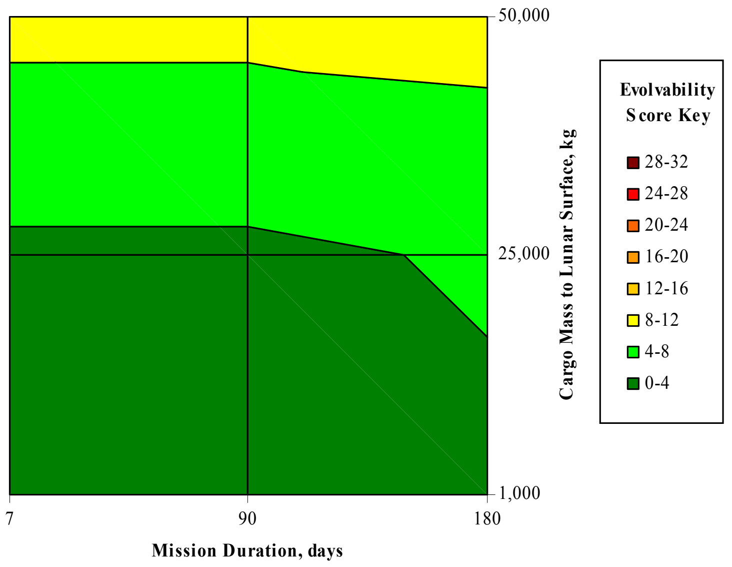

Figure 6. LSR architecture (FLO) evolvability score contour plot for four crew members.

D. Selecting the Most Evolvable System

The contour plots shown in Fig. 5 and Fig. 6 are useful when the behavior of the system is of interest, but sometimes more concise metrics are also desirable. How does one select the "most evolvable system?"

Unfortunately, it is not necessarily useful to present a system's overall evolvability as a single number- this is better done through contour plots similar to those shown above. This is because a single metric cannot capture the sensitivity of an architecture's capability to evolve in the direction of required evolution. If a single metric is desired, a single evolved state must be defined. This allows the decision-maker to compare the candidate architectures based on their ability to evolve to a particular evolved state. Because the evolved state is defined, each architecture should receive a single evolvability score. The architecture with the lower score is better suited to evolve to the extended mission.

Take, for example, the lunar case study investigated throughout this paper. Suppose the customer desired to select the architecture most capable of evolving to a lunar mission that sends a crew of six to the lunar surface for 90 days, along with the capability of sending 25,000 kg of cargo to the lunar surface each year. Comparing the evolvability of these two architectures to this new mission yields the results shown in Fig. 7. For this example, the LSR architecture (FLO) is more evolvable to this particular mission because it has the lower score.

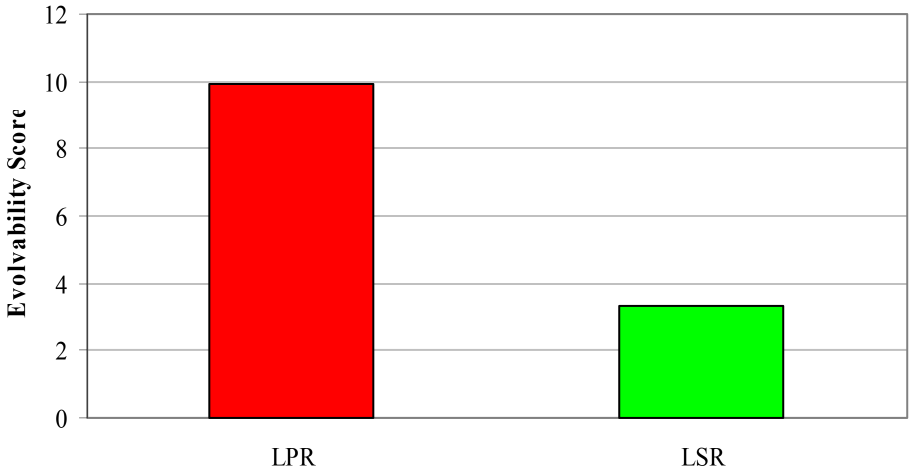

Figure 7. Evolvability score comparison between LPR architecture (OASIS) and LSR architecture (FLO) for a specific evolved lunar surface mission.

This particular example underscores the necessity of defining the evolved state when stating that one architecture is more evolvable than another. When asked in an informal manner which system they felt was more evolvable, seven of the students in the SSDL said OASIS is the most evolvable, while four students said FLO is the most evolvable. Figure 7 clearly indicates that FLO is more evolvable to the stipulated evolved state, so why did over half the group say that OASIS was more evolvable?

Individuals often have preconceived notions of what types of systems are evolvable. Without the structured environment described above, it can be difficult to select the architecture that is most evolvable to the specific evolved mission under consideration. Lacking the guidance of a structured methodology, an architecture's ability to evolve to unrelated missions may cause it to look more evolvable than it actually is. Take, for example, the OASIS architecture. Its modularity and the location of a station at EML_1 enable OASIS to easily evolve to many missions within Earth's neighborhood (i.e. servicing satellites or stations at other libration points, acting as an in-space storage depot, etc.). Such capability would be much more difficult to achieve through evolving FLO. In this particular case study, however, the goal was to determine which architecture could better evolve to a more ambitious lunar campaign. Here, the other capabilities of OASIS are not important. What counts is its ability to evolve to ambitious lunar missions- something FLO is more capable of doing.

The problem, therefore, generally lies in considering an architecture's evolvability to states other than those of interest. The methodology described in this paper alleviates this problem by providing a structured framework for organizing information and ranking system evolvability.

V. Conclusion

Current events have made evolvability a sought after attribute. In this paper it was proposed that evolvability may be measured as a figure of merit, allowing us to finally ask the following question: is evolvability really a desirable property? By considering evolvability among a host of other figures of merit (such as cost, reliability, performance, etc.) the decision-maker can decide if they can afford a highly evolvable system.

In this paper, a quantitative procedure for evaluating system evolvability was outlined and applied to an example case study. A refined definition and functional breakdown of evolvability provide a structured framework in which this metric may be discussed and measured. The proposed methodology allows for the computation of numeric evolvability scores indicating a system's evolvability. With additional refinement, the scope and reliability of this approach may be improved. Furthermore, modification of the EDSA should also allow for increased flexibility in meeting the needs of the customer.

Acknowledgments

The authors thank the members of the Georgia Tech Space Systems Design Lab who participated in developing the difficulty matrices used for the case study.

References

[1]Bush, G.W., "A Renewed Spirit of Discovery," NASA Headquarters, Washington, D.C., January 14, 2004.

[2]Aldridge, E.C. et al., *A Journey to Inspire, Innovate, and Discover,* President's Commission on Implementation of United States Space Exploration Policy, 2004.

[3]*SMC Systems Engineering Primer and Handbook*, Space and Missiles Center, U.S. Air Force, 2004.

[4]Rowe, D., and Leaney, J., "Evaluating Evolvability of Computer Based Systems Architectures- An Ontological Approach," *IEEE Proceedings ECBS*, March 24-28, 1997, Monterey, CA, pp. 360-367.

[5]Rowe, D., Leaney, J., and Lowe, D., "Defining Systems Evolvability- A Taxonomy of Change," *IEEE Proceedings ECBS*, March 30-April 3, 1998, Jerusalem, Israel, pp. 45-52.

[6]Percivall, G.S., "System Architecture for Evolutionary Development," *Proceedings of the 4th Annual Symposium of the National Council on System Engineering*, August 1994, San Jose, CA, pp. 571-575.

[7]Isaac D., and McConaughy, G., "The Role of Architecture and Evolutionary Development in Accommodating Change," *Proceedings of the 4th Annual Symposium of the National Council on System Engineering*, August 1994, San Jose, CA, pp. 541-545.

[8]Bunge, M., *Treatise on Basic Philosophy, Volume 3, Ontology I: The Furniture of the World*, Reidel, Boston, MA, 1977, Chaps. 3, 5.

[9]Bonfatti, F., and Pazzi, L., "Modeling Object Complexity and Behavior: Towards an Ontological Paradigm," *IEEE Proceedings CompEuro*, May 13-16, 1991, Bologna, Italy, pp. 844-849.

[10]Wand, Y., and Weber, R., "An Ontological Model of an Information System," *IEEE Trans Software Engineering*, Vol. 16, No. 11, 1990, pp. 1282-1292.

[11]Wand, Y., and Woo, C.C., "An Approach to Formalizing Organizational Open System Concepts," *ACM Proceedings Organizational Computing Systems*, November 5-8, 1991, Atlanta, GA, pp. 141-146.

[12]Klir, G.J., *Architecture of Systems Problem Solving*, Premium Press, New York, NY, 1985, Chap. 6.

[13]Troutman, P. A., et al., "Orbital Aggregation and Space Infrastructure Systems (OASIS)," *World Space Congress 2002*, October 10-19, 2002, Houston, Texas.

[14]*Orbital Aggregation & Space Infrastructure Systems (OASIS)- Preliminary Architecture and Operations Analysis*, *FY2001 Final Report*, NASA Revolutionary Aerospace Systems Concepts. June 2002.

[15]Asker, J.R., "U.S. Draws Blueprints for First Lunar Base," *Aviation Week and Space Technology*, August 31, 1992, pp. 47-51.

[16]Day, D.A., "The Last Lunar Outpost," *Spaceflight*, Vol. 46, No. 10, October, 2004, pp. 399-403.

[17]*IEEE Standard Glossary of Software Engineering Terminology*, Institute of Electrical and Electronic Engineers, IEEE 610.12-1990, New York, NY, 1990.

[18]Buede, D.M., *The Engineering Design of Systems: Models and Methods*, John Wiley & Sons, Inc, New York, NY, 2000, pp.436.