

Time-Expanded Decision Network: A New Framework for Designing Evolvable Complex Systems

Matthew R. Silver* and Olivier L. de Weck.†
Massachusetts Institute of Technology, Cambridge, MA 02139

This paper describes the concept of Time-Expanded Decision Networks (TDN), a new methodology to design and analyze flexibility in large-scale complex systems. This includes a preliminary application of the methodology to the design of Heavy Lift Launch Vehicles for NASA's space exploration initiative. Synthesizing concepts from Decision Theory, Real Options Analysis, Network Optimization, and Scenario Planning, TDN provides a holistic framework to quantify the value of system flexibility, analyze development and operational paths, and identify designs which can allow managers and systems engineers to react more easily to exogenous uncertainty. TDN consists of five principle steps, which can be implemented as a software tool: 1. Design a set of potential system configurations 2. Quantify switching costs to create a "static network" that captures the difficulty of switching among these configurations 3. Create a time-expanded decision network by expanding the static network in time, including chance and decision nodes 4. Evaluate minimum cost paths through the network under plausible operating scenarios 5. Modify the set of initial design configurations to exploit high-leverage switches and repeat the process to convergence. Results can inform decisions about how and where to embed flexibility in order to enable system evolution along various development and operational paths.

Nomenclature

C_{LCi}	=	estimated lifecycle cost of the i-th system configuration (design)
C_D	=	cost of DDT&E
C_F	=	fixed recurring cost per time period
C_V	=	variable cost as a function of demand
$C_{SW}(A, B)$	=	cost of switching from A to B
$C_{DEV}(B)$	=	DDT&E of the new system configuration B
$C_R(A)$	=	cost of retiring the old system A
$D_{i,j}$	=	demand for system configuration i during period j
r	=	discount rate
T	=	number of time steps considered
ΔT	=	time step

I. Introduction

There is increasing recognition that *adaptability*¹ is a critical determinant of the long-term effectiveness and, in relevant cases, profitability of complex technical products and systems. Many complex systems exhibit high degrees of architectural lock-in² making them difficult to adapt and manage in the face of environmental change and uncertainty, and by extension hindering adaptability. Lock-in has been the subject of much research in various fields including engineering (dominant design)^{3,4}, economics and politics and other topics concerned with systems design and innovation management. The exact causes of lock-in are still debated⁵. They are generally acknowledged to be multidimensional, taking root in socio-technical issues such as network externalities^{6,7}, the cost of learning new technologies⁸ and operating procedures, the risks associated with switching to new systems, the trade-off between operating slightly inefficient fielded technologies and developing new technologies⁹, and broader issues associated

* Research Scientist, Dept. of Aeronautics & Astronautics, 33-409, AIAA Student Member, mrsilver@mit.edu

† Associate Professor, Dept. of Aeronautics & Astronautics, Engineering Systems Division, 33-410, AIAA Senior Member, deweck@mit.edu

with political and organizational inertia¹⁰. For large-scale technical systems, these factors can be grouped into the general concept of *switching costs*, which are weighed—whether formally or informally—against the benefits of switching to new systems, products or operating procedures. Switching costs can be real dollars, or quantifiable costs associated with personnel considerations, political implications, or the time it takes to switch. They are further associated with what might be called *switching risk*¹¹.

Increasing the adaptability or flexibility of a system demands lowering future switching costs before the environmental changes which demand a switch arise. Suh has used the concept of switching costs to find where to embed flexibility in automotive platforms¹². We argue that system designers must mitigate the causes of architectural lock-in at the design stage, before systems are fielded. Much recent research has addressed the need to design for flexibility and methods to design for it. Real Options Analysis for example, is a method of valuing system flexibility by framing system design in terms of stock-option theory.[‡] Similarly, the concepts of Spiral or Evolutionary Development emphasize the need for continual development and improvement of large systems rather than all-at-once development and operation. At their core, these methods seek either ways of systematically and strategically identifying the benefit of various kinds of flexibility and introducing it at the design stage, or delaying critical design decisions until environmental uncertainties are resolved. The differences between adaptability in the systems themselves versus adaptability in the system development process have recently been clarified by Haberfellner and de Weck¹³.

Building on these studies, we introduce a new methodology and associated modeling tools to design “evolvable” complex systems. Given their size and long-life cycles, complex technical systems are best characterized as operational organizations evolving in the face of exogenous uncertainty, able to take various development paths when confronted with environmental change, rather than fixed sets of technical components. In a nod to information theory, the determinant of a system’s “evolvability” is a function of the number of possible states that it might take, rather than the state it is in, with the crucial caveat that alternating between states incurs some kind of *switching cost*. TDN presents a method to clearly and rigorously represent feasible paths and, to the extent possible, quantifying expected switching costs. These paths and costs can then be represented on a directed network with arc-costs representing cost-elements incurred during system operation, and expanded through a life-cycle using standard methods from network theory, called a time-expanded network. Once formulated, designers can use the time-expanded decision network to conduct scenario planning and iteratively design more evolvable complex systems.

As noted, while the sources of architectural lock-in are multi-dimensional, from an economic perspective it stems from high capital expenses (capex), whether monetary or not, associated with switching to new systems. Framed this way, the problem becomes fairly simple: locked-in will occur if the cost of switching from one system configuration to another exceeds the expected benefits, discounted at some rate and assuming a level of risk (Simon).^{14,§} In order to design for adaptability, we must lower switching costs. This insight leads directly to two important questions: (1) How can we quantify switching costs? (2) Once quantified, how can we design systems to diminish the appropriate elements of switching costs in order to improve life-cycle adaptability and, by extension, performance? The TDN methodology described in the next section addresses these questions explicitly. Section III then applies the method to the specific example of launch vehicle development and operations under demand uncertainty.

II. Time Expanded Decision Network (TDN)

TDN Methodology Overview

The TDN methodology involves identifying and quantifying switching costs, creating an optimization model based on the concept of a time-expanded network, and running this model under probabilistic or manually generated demand scenarios to identify optimal designs. TDN is, in effect, a method to run “quantitative scenario planning” for

[‡] For a good overview of Real Options Theory, see: “Real Options in Capital Investment: Models, Strategies, and Applications.” Edited by L. Trigeorgis

[§] The problem is actually a bit more complicated, as uncertainty in the operating environment creates a further incentive not to switch even if the benefit is equal or slightly above the cost of switching. This is due to the fact the environment could change again, making the switch unnecessary, so there is a “benefit of waiting.” This phenomenon has been called “sunk cost hysteresis” by economists and has been studied and modeled extensively¹¹. The phenomenon is outside the scope of this paper, but could be included into the methodology later.

system design, in which system responses are encoded through a time-expanded network. The crux lies in identifying how and where switching costs are incurred and how these can be lowered in order to maximize adaptability and, perhaps more importantly, *determining exactly how much one should be willing to spend to lower these switching costs at the development stage.*

The TDN methodology consists of the following five steps (Fig. 1):

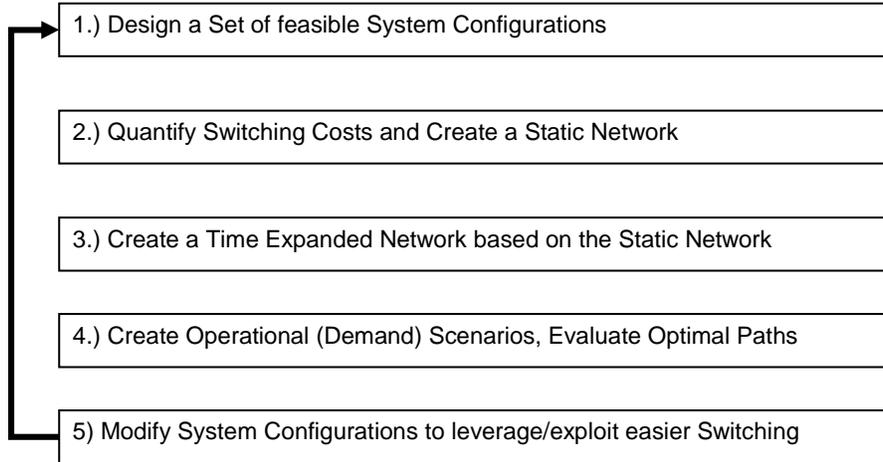


Figure 1: Flow of the 5-step TDN Methodology

These steps are described in more detail below, using a generic example of designing a system with three potential instantiations (configurations) and then implemented in the following section (Section III) in the context of launch vehicle design and operations.

Step 1: Design a Set of feasible System Configurations

The first step involves creating a family or set of independent designs. This can be a core platform with options to expanded or wholly unrelated point designs. Once conceptual design configurations have been refined, the basic elements of life-cycle cost for each point design can be estimated. These are:

1. DDT&E (Design, Development, Testing and Evaluation)
2. Fixed Recurring Cost during Operations (independent of demand)
3. Variable Recurring Cost during Operations (dependent on demand)

DDT&E generally includes design, analysis, testing of subsystems, system integration, system tests, and construction of any ground equipment and facilities¹⁵. *Fixed recurring costs* are those incurred regardless of demand and/or expected changes in the operating environment, including labor, facilities, overhead. For large, complex systems, labor is usually a driver for fixed recurring cost. *Variable recurring cost* is a function of the demand during a given year or time period. It includes operating costs, production materials, and variable labor costs, among other elements. Models of variable recurring cost often take into account benefits of learning, which reduces cost per unit as a function of cumulative production levels.

Regardless of how these three traditional cost-elements are modeled, they are generally used to estimate life cycle cost, C_{LCi} , of a given design i , as a function of the number of periods T , and the demand in each period, $D_{i,j}$:

$$C_{LCi}(D,T) = C_{Di} + C_{Fi} \times T + \sum_{j=1}^T C_{Vi}(D_{i,j}) \quad (1)$$

Note that discounting (time value of money) could be included, which leads to:

$$C_{LCi}(D, T) = C_{Di} + \sum_{j=1}^T \frac{C_{Fi} + C_{Vi}(D_{i,j})}{(1+r)^j} \quad (2)$$

Step 2a: Quantifying Switching costs

Once a set of system configurations (e.g. a family of potential launch vehicles) has been designed, and the three traditional cost-elements (DDT&E, fixed, variable) established for each point design, we must now identify the cost of switching between these system configurations. As noted, switching costs are difficult to quantify because they combine direct production costs as well as indirect organizational and political costs associated with learning and changing established operating procedures. For our current purposes we assume that switching costs comprehend two principal elements:

1. Technical cost associated with altering and producing a new configuration and/or technology
2. A premium for organizational change

The first element includes the cost of designing, building, and testing the new system (C_{DEV}) and the cost of retiring the old system (C_R). Thus, the technical cost of switching from configuration A to B is the cost of developing B, plus the cost of retiring A:

$$C_{SW}(A, B) = C_{DEV}(B) + C_R(A) \quad (3)$$

The key assumption is that A and B will not operate concurrently, however, there is *commonality* between A and B—that is, if A and B share common subsystems or are based on common platforms¹⁶— C_{DEV} and C_R must be estimated using more detailed analysis of which subsystems are re-used and which must be changed. The premium for organizational change is optional, but can be a linear function of the number of people involved in designing and operating the technology. As a whole, adequately quantifying switching costs is a difficult problem which merits further study. One example is presented below, in the context of launch system design (Section III).

Step 2b: Creating the Static Network

Once switching costs between system configurations have been estimated they can be placed in a matrix, with original system configurations (“from”) on the rows and new system configurations (“to”) as the columns. To take a generic example, imagine three possible point designs for a given system: A, B, C. The costs of switching from each to each, $C_{SW}(x, y)$ is placed in a matrix (Table 1). Numbers at each cell in the matrix represent the cost of switching from the *i*th row (configuration) to the *j*th column (configuration). The matrix is not necessarily symmetric.

Table 1: Example switching costs between system A, B and C

Switching Cost From	To A	To B	To C
A →	0	$C_{SW}(a, b)$	$C_{SW}(a, c)$
B →	$C_{SW}(b, a)$	0	$C_{SW}(b, c)$
C →	$C_{SW}(c, a)$	$C_{SW}(c, b)$	0

This matrix is formally equivalent to what is called a *node-node adjacency matrix* in graph theory^{17,18} and network flow theory more generally. That is, the matrix of switching costs defines a network whose nodes are the point-designs (configurations) and whose arc weights are the switching costs according to Eq. (3). The resulting graph is represented in Figure 2.

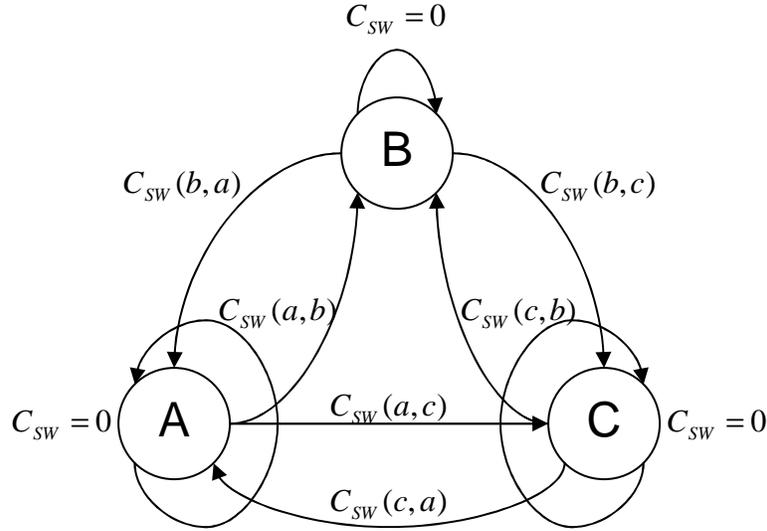


Figure 2: Graph representation of the switching cost matrix between A, B and C

We call the representation above a “static network.” It covers all possible switches between the three systems at any given point in time. Because all n nodes in the graph are connected to all nodes as well as to themselves, as a rule the number of arcs in this static network will be n^2 . It is important to note that the switching cost associated with staying in a given system configurations (at the same node in the static network) is zero. This does not mean that the cost for keeping the system operating in that configuration is zero—there are still fixed and variable recurring costs.

Step 2c: Completing the Static Network

To frame the system’s life-cycle as a holistic dynamic network flow problem we must add costs associated with developing and operating the system, see Eq. (1, 2). Then we must add the element of time to each link. The costs associated with running each system configuration (C_D, C_F, C_V) on the static network above can be coded into the network as follows:

Let N represent the static switching network defined above (Fig. 2) consisting of a set of nodes (point designs) $P_0, P_1, P_2, \dots, P_n$ and oriented arcs associated with switching costs, $C_{SW}(i,j)$. Let P_{ij} represent the directional arc from node P_i to node P_j . Associate the self-loops (that is, all arcs from a node to itself P_{ii}) with the sum of fixed recurring and variable recurring cost elements for that configuration, i .

$$P_{ii} = C_{Fi} + \sum_{j=1}^T C_{Vi}(D_{i,j}) \quad \text{(undiscounted)} \quad (4)$$

Now create a source node S and a sink node Z and connect them both to all nodes with appropriate directional arcs. Associate all arcs P_{sj} flowing from the source to the point designs with the development cost of design j , C_{Dj} . This represents the DDT&E cost of the initially chosen configuration. Associate all arcs flowing from point designs to the sink P_{jz} with the cost = 0.** The result is a static network with all information relevant to the life-cycle cost, as a function of demand, encoded in the arcs.

** Making all sinks arcs = 0 makes the assumption that there are no retirement costs associated with the system at the end of its lifecycle. Retirement costs could be encoded into the network at these arcs.

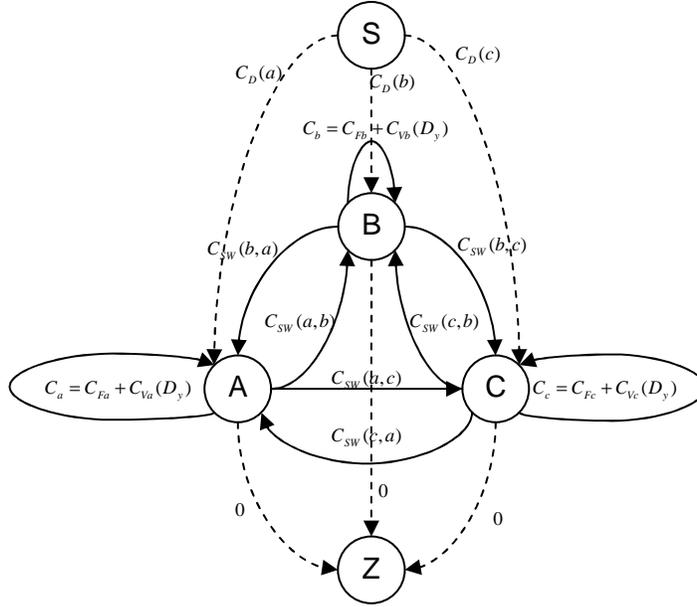


Figure 3: Complete Static Network representing switching and staying costs, as well as development costs. Development is represented by the dotted line from source S to all nodes. Retirement is represented by the dotted line from all nodes to the sink Z.

Step 3: Creating the Time Expanded Network

The goal of the TDN is to capture the cost of operating and switching system configurations through time, seeking least-cost paths through the network as a function of changing operational requirements such as demand. The network above captures all possible development paths during a system’s life-cycle assuming the set of configurations represented in the initial static network, but it is not *dynamic*—that is, it does not include the element of time. The introduction of time into a directed network transforms it into a *minimum dynamic network flow problem*. The generic problem involves finding min or max-cost flows from a source, S , to a sink Z , in a directed network with transversal costs, c . *Dynamic* network flow problems further associate each arc with two variables: traversal cost, c , and a traversal time t . Often the reformulated objective is to find the maximum flow that can be sent through the network in time T , where costs represent maximum capacity. A solution to the maximal dynamic network flow problem was first presented by Ford and Fulkerson¹⁷ (Ford, and Fulkerson 1958) and subsequently elaborated upon in a host of different practical applications.

Dynamic network flow problems can be solved by reformulating the dynamic network into a *time-expanded static network*, and then solving a traditional max or min cost flow problem on the resulting static network (Fulkerson 1958, others). A time expanded network splits the problem into T time periods, with each node in the original network duplicated every time period. Arcs connect the nodes depending on traversal period, and the new network contains only cost information for each arc. Methods to reformulate static networks into time-expanded networks are well established (see Ford, Fulkerson¹⁷, pp. 146). We first divide the life-cycle into T time periods and associate each arc with a traversal time, t . For simplicity and generality, let us assume here that each arc in the network (switching or staying) takes exactly one time period ΔT to traverse.^{††} Each node in the static network is duplicated in each time period, and arcs between these nodes are expanded through the new network based traversal times, t .

^{††} Alternations to this assumption can easily be made based on specific modeling needs. Sometimes, for example, switching to a new system will take multiple time periods. These kinds of issues can be addressed in different ways by, for example, by changing the length of each time period or changing the cost of switching to take account of the fact that it is really spread of multiple time periods (years). For our modeling purposes in Section III, we assume that one time period is one calendar year, and that it corresponds to making major development decisions in context of the yearly budget reviews in the US government.

Figure 4 expands the static network created by hypothetical system configurations A, B, C, into a time-expanded network over 3 time periods, T1, T2, T3. It is assumed the traversal time associated with each arc is exactly one time period ΔT . Flow starts at the start node, S, and ends at the end node, Z.

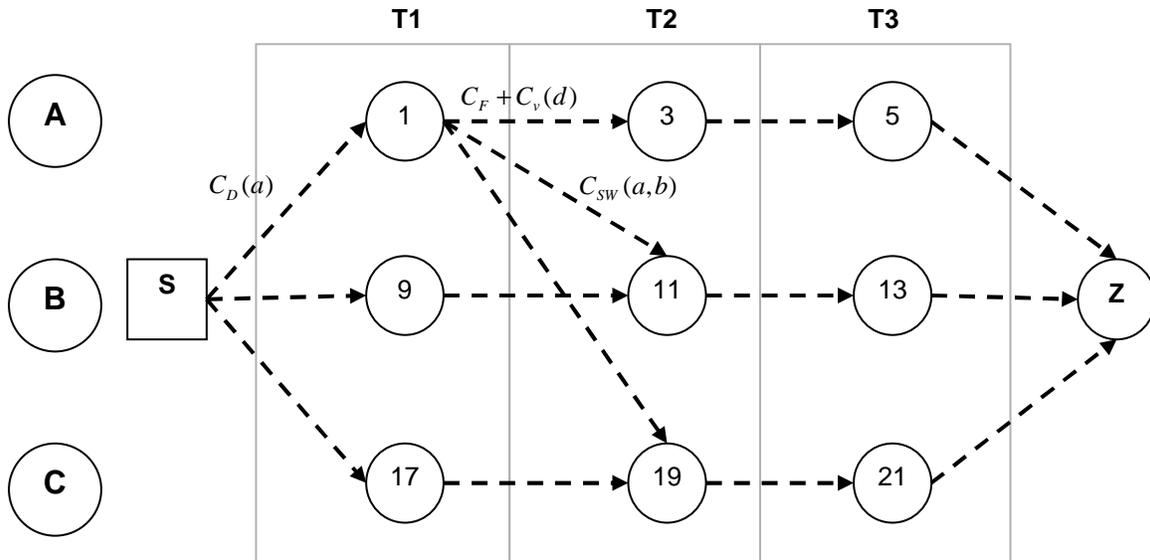


Figure 4: Example Time Expanded Decision Network

Figure 4 represents the development alternatives and costs over a life-cycle, given three system configurations, A, B, C, and our four elements of cost (C_D, C_F, C_V, C_{SW}), however, one critical element is still missing: if a path through the network includes a switching arc, the “operating arc” during that time period will have been avoided. In other words, a min-cost flow through the above network can selectively chose to avoid periods of high-demand by switching before them. In reality, an existing system will continue to operate to meet existing demand while a new one is being developed.

We therefore introduce a final distinction, taking a cue from decision theory, between chance nodes and decision nodes, and thereby decoupling operating costs from switching periods. This is done by splitting each time period with chance nodes at the beginning, and decision nodes at the end. Fixed and recurring costs are incorporated into the sub-arcs following the chance nodes at the beginning of each time period (let us call them *chance arcs*). Switching costs on the other hand are incorporated into the arcs following the decision nodes at the end of each time period (let us call them *decision arcs*). The chance arcs represent the costs of operating the system at a level of demand, D , in time period T_i . The resulting time-expanded network is as follows (Fig. 5):

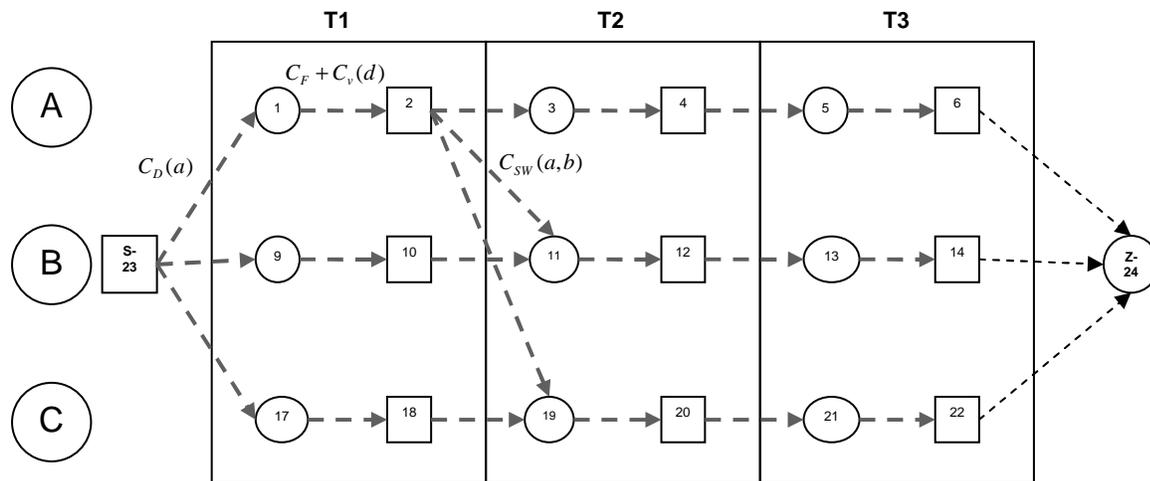


Figure 5: Time Expanded Decision Network. Chance nodes are circles, decision nodes are squares.

Step 4a: Create Operational Scenarios (Modeling Demand)

It is now possible to introduce operational scenarios over time, by changing the cost of traversing the chance arcs as a function of operational demand and by creating various demand scenarios. Demand scenarios can be discrete, involving levels of demand at each period, or probabilistic. A simple way to examine the space of demand possibilities is to create a number of discrete demand profiles that are likely to occur over time (i.e. steadily rising demand, constant demand, falling demand, and a limited combination of the three) and applying these to the network. Another way involves modeling using Geometric Brownian Motion (GBM)¹⁹, and applying multiple combinations of demand scenarios (Fig. 6).

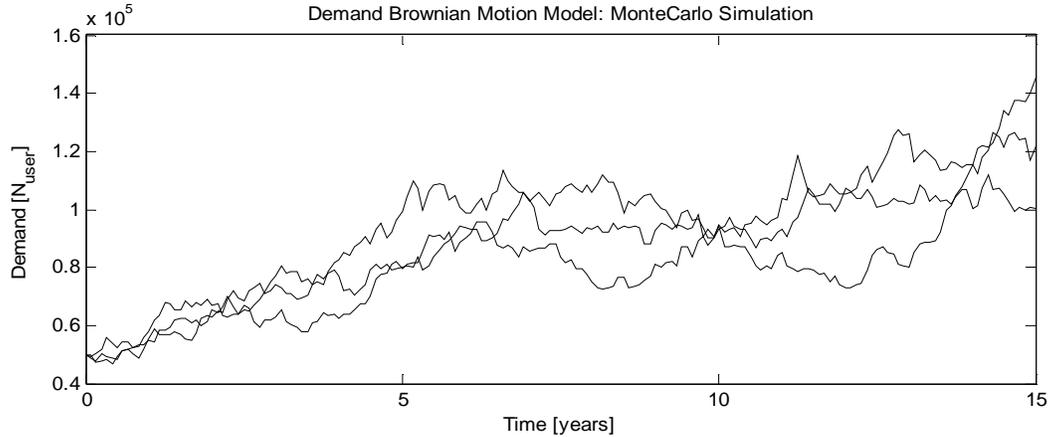


Figure 6: GBM model of uncertain demand, $\Delta T = 1$ month, $D_0 = 50,000$, $\mu = 8\%$ p.a., $\sigma = 40\%$ p.a. – 3 scenarios are shown (de Weck et al. 2004)

The important point with respect to the TDN methodology is that these costs can be calculated using separate and standard cost models and then used as inputs to the time-expanded network. That is, the total cost of operating in each time period can be computed as a function of the demand scenario and the particular present state (configuration) of the system, and then *these costs can be used as the cost of traversing the chance arc in that time period.*

Step 4b: Finding the Optimal Path

The ultimate goal for the simulation is to find the set, or sets, of designs that minimize life cycle cost under various demand scenarios. By extension, we want to identify exactly where lowering switching costs may maximize adaptability. The set may, or may not include more than one point design, depending on the relevant ratios of the four cost elements (C_D, C_F, C_V, C_{SW}) and the demand at each time period. In other words, this may or may not involve switching from one design to another during the lifecycle.

Once the costs of operation are coded into the chance arcs of the time-expanded network, finding least-cost paths involves implementing standard network optimization algorithms. If no loops exist in the network, as in Fig. 5, nodes can be order topologically—that is, ordered in terms of how far they are from the source. When this is the case, a simple reaching algorithm¹⁸ can be used to find the shortest path. The reaching algorithm is used in this paper. For each operational scenario a different path through the TDN may be optimal. We are interested in identifying those configurations $A, B, C \dots$ that are most often chosen as initial configurations and those switches $i \rightarrow j$ that are selected most often to switch among these configurations.

Step 5: Modify System Configurations (Iterative Design)

Once formulated, the time-expanded decision network model is a powerful tool to help formulate and refine initial point designs as well as identify opportunities for commonality and real options among these configurations. A real option is simply a feature that is embedded in an initial design configuration to lower future switching costs. The TDN can be used in many different ways to creatively identify point designs that may lower future operating costs.

By combining myriad demand scenarios and network optimization algorithms, designers can selectively tweak elements of point designs, increasing and decreasing commonality, to see exactly how this affects least-cost operating and development paths through the network. As will be illustrated in the example below (Section III), this information can be used to exploit high-leverage switches, that can serve to open vast new operating spaces that would never have been exploited had switching options not been incorporated into previous designs.

III. Application of TDN to the Design of Heavy Lift Launch Vehicles

The TDN method described above has been programmed as a functional Matlab tool and applied to the problem of designing an “evolvable” Heavy Lift Launch System (HLLV) for NASA’s Space Exploration Vision.^{††} For this problem, four different initial vehicle configurations were created— one clean sheet design, one based on Evolved Expendable Launch Vehicles (EELV) and two based on Shuttle-Derived vehicle alternatives—and related cost elements and switching costs were computed. The methodology was run using 10 plausible scenarios for Lunar and Mars exploration campaigns²⁰, and total life-cycle costs were calculated. Initial results demonstrated preferred vehicle configurations, and iterative methods quantified the exact value of lowering switching costs under a uniform probability distribution.

Step 1: Initial System Configurations

The model was implemented using four point designs of interest to NASA during the heavy lift launch decision (Fig.7). An N² diagram showing the components of the launch vehicle models is shown in the Appendix.

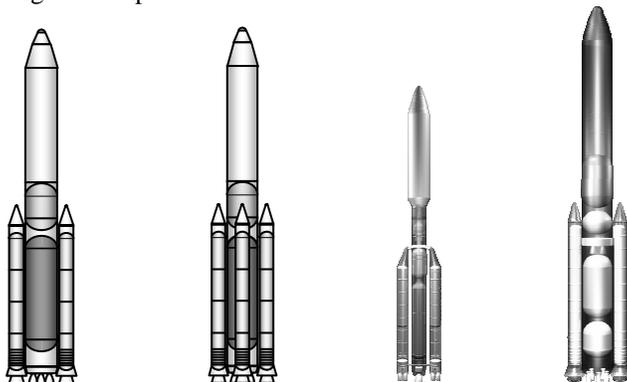


Figure 7: Four launch vehicle point designs used in the TDN. From left to right: Shuttle Derived A (1); Shuttle Derived B (2); EELV derived C (3); Clean Sheet D (4)

1. A: Shuttle Derived Inline Heavy Lift; 3 Space Shuttle Main Engines; 2 four segment solid rocket boosters; *Performance: 80 metric tons to low earth orbit*
2. B: Shuttle Derived Inline Heavy Lift; 3 Space Shuttle Main Engines; 4 four segment solid rocket boosters; *Performance: 115 metric tons to low earth orbit*
3. C: Evolved Expendable Launch Vehicle (EELV) derived rocket; Delta V core; 2 Zenith boosters; *Performance: 62 metric tons to lower earth orbit*
4. D: Clean Sheet Heavy Lift Vehicle; *Performance: 105 mt to low earth orbit*

After performance and risk were estimated for each configuration, the first step of the TDN process involved estimating development costs for all four vehicles as well as fixed and recurring operating costs as a function of demand. This was done by NASA and supplemented by the authors using traditional cost-estimating tools. For more information on this process and underlying assumptions of this case study see (Silver 2005)²⁰.

Step 2: Calculating Switching Costs and Create Static Network

The next step involved calculating switching costs between the vehicle configurations shown in Fig. 7. As noted above, this depends crucially on the amount of commonality between the systems and any real options (reserved

^{††} For an N-squared diagram of the software program created to run the TDN methodology, see *Appendix B*.

interfaces, extra margins...) embedded in them. Systems can share subsystems at different levels. For example, Shuttle-derived vehicles and EELV derived vehicles might be designed to use common large-scale subsystems such as launch pads, with minor alterations. At a deeper level, two shuttle derived vehicles might share almost all first order subsystems—launch pads, manufacturing facilities, rocket boosters, main engines—with alterations only to specific second-order subsystems—i.e. the external fuel tank. To calculate switching costs, then, we must break the system into its constituent subsystems to identify commonality at the appropriate level.

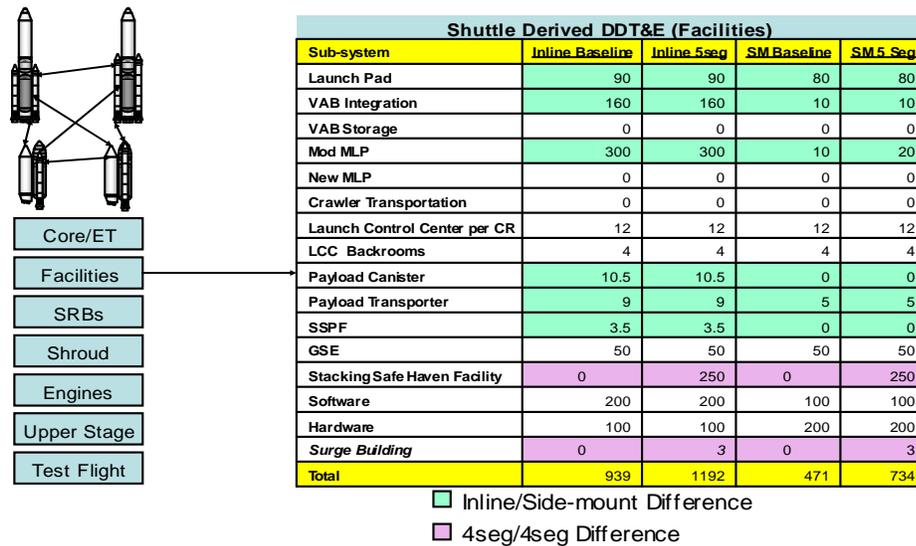


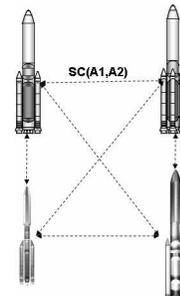
Figure 8: Estimating Switching Cost by decomposing subsystems.

To take one example of how point designs can involve commonality at a level deeper than the major subsystem level, imagine four shuttle-derived point designs (Fig. 8): A side-mount configuration with four segment solid rock boosters (SRB); A side mount configuration with five-segment SRBs; An inline vehicle with 4-segment SRBs; and an inline launch vehicle with 5-segment SRBs. Each of these system pairs has commonality in a different way—some share the same solid rocket boosters, others are inline. Calculating switching costs between these systems demands examining how their common elements affect subsystem design. Figure demonstrates how this can be done by decomposing major subsystems and examining how they affect facility design. In this case some facilities are common and some facilities must be alternated to create either four or five segment SRBs, and similarly some must be altered to create inline or side mount vehicles. These elements are not mutually exclusive and therefore the switching costs between each system must take into account what is already built and what has yet to be built.

In the TDN simulation the resulting switching costs between systems 1,2,3,4 are presented in Table 2 below.

Table 2: Switching Costs (\$M) between four launch vehicles

Switch Costs	1	2	3	4
1	0	2285	4600	8900
2	0	0	4600	8900
3	6718	8848	0	8900
4	6718	8848	4600	0



Step 3: Create Time-Expanded Network and Demand Scenarios

Once the four cost elements were determined, we created the time expanded network (Figure). In this case we had *four system configurations and divided the time horizon it in to four periods*. Ten demand scenarios were created.

Modeling demand is a complex process and can be done in many different ways. Demand in our case involves how many tons of payload need to be shipped to low-earth orbit (LEO) per time period^{§§} and how many launches this would necessitate. NASA is principally concerned with returning to the Moon and possible sending missions to Mars, therefore, we had to translate variations in kind and timing of Lunar and Mars missions to launch-vehicle demand (see Silver 2005). Next, we created ten scenarios based on how many lunar and mars missions might be launched during each time period and translated this to different numbers of launches for each vehicle and, by extension, different patterns of operations costs along the chance arcs in the time-expanded network.

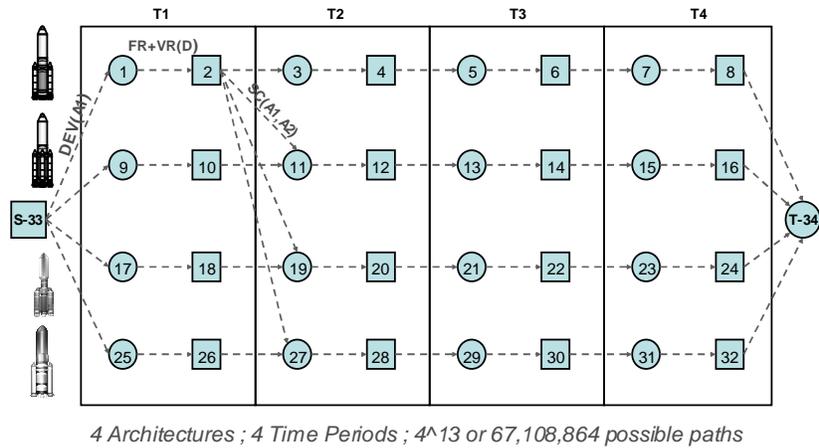


Figure 9: The time expanded decision network (TDN) for the launch vehicle case

Step 4: Path Optimization

Once the demand scenarios we created and the resulting operations costs coded into the network we ran the network optimization algorithm under each scenario. The result was ten different shortest paths through the life-cycle, as a function of each demand scenario. We found that only two vehicle configurations (1 and 3) were ever used and that, moreover, no switches were ever used. Figure 10 illustrates the two shortest paths, and Figure shows the total life-cycle cost predicted for each scenario.

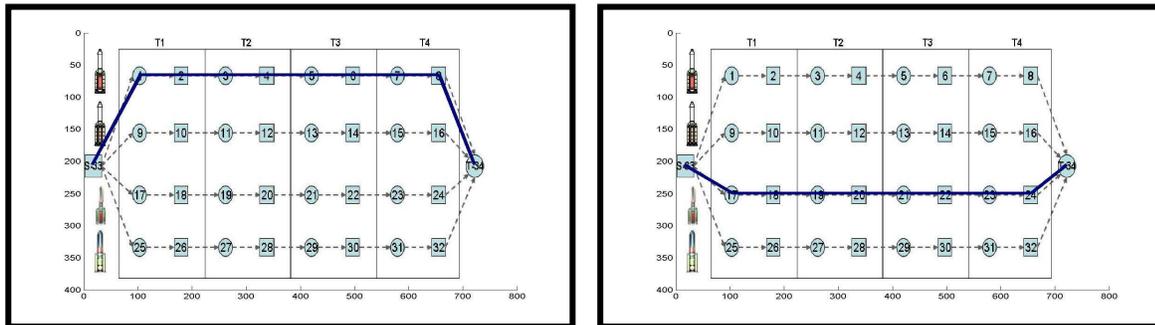


Figure 10: Two resulting shortest paths through the life-cycle

^{§§} A lunar mission requires approximately 130 metric tons to lunar orbit, while a Mars mission requires on the order of 700 metric tons to be delivered to Low Earth orbit.

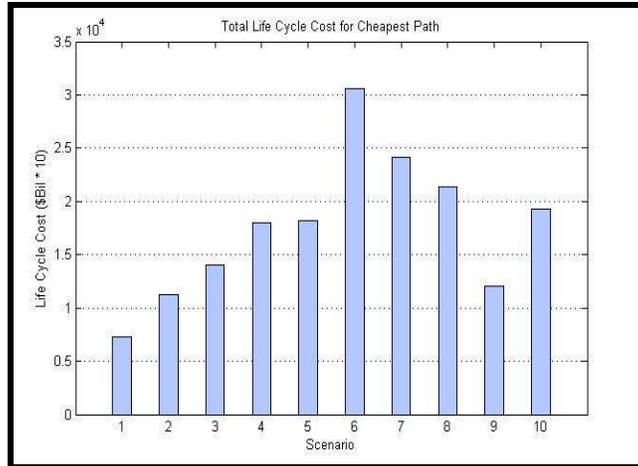


Figure 11: Total life cycle costs for all ten scenarios

Step 5: Iterative Design

The results demonstrated that two vehicles (configuration 1 and 3, respectively) were optimal under different operating regimes and immediately raised the question: *Would it be worthwhile to decrease the switching costs between these two vehicles and, if so, how much should we be willing to pay to do this?* We therefore systematically lowered the switching costs between these two vehicles manually, in order to determine (a) at what switching costs did switching start to occur under each demand scenario and (b) what were the savings, if any, in total life-cycle cost.

More specifically, we systematically lowered the cost of switching from the EELV-derived vehicle (small launch vehicle used initially for Moon missions) to the shuttle-derived SRB vehicle (larger vehicle used for Mars missions later) from \$4 billion dollars to \$1 billion to determine how much this would change the path taken over the life-cycle of the system. The results are summarized in Figure . We found that if the switching cost is set to \$4 billion, one switch is made in scenario 8 after 2 time periods—this is a scenario where we encounter few lunar missions initially and many Mars missions later. As the switching cost is lowered further, switches also occur in other scenarios, see Fig.12.

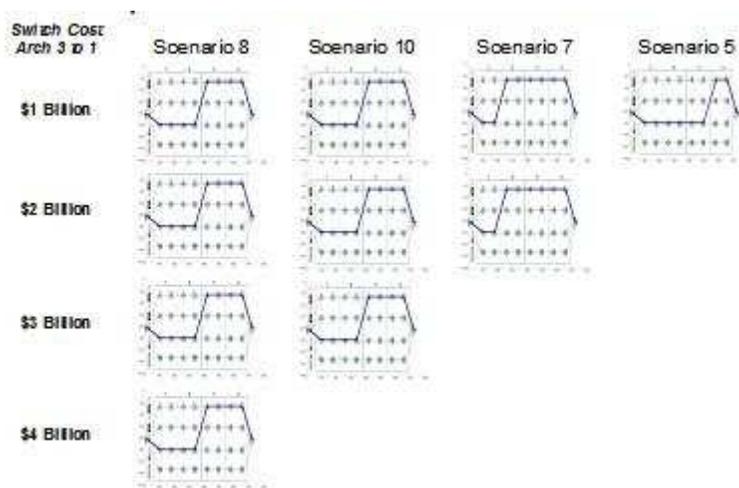


Figure 12: Switches as a function of switching cost level between \$1-4 billion

Calculating the Benefit

What would be the benefit of lowering switching costs in this way? The answer depends on the exact scenario that ends up occurring in the future. The simulations demonstrated that if scenario 8 did occur, that a switch between the vehicles occurs even at a relatively high switching cost. This implies that if scenario 8 were to occur, the savings of lowering switching costs between these vehicles would be high. Another way to present this, however, is to determine how much is saved in terms of expected-cost if each scenario is equally likely—in our case there are ten scenarios, so each scenario would have a 10% chance of occurring.

Figure 13 illustrates the total life-cycle cost as a function of the switching cost between the vehicles, both for scenario 8 in particular and for the case where every scenario is equally likely. *SC_Normal* is the original case (with switching costs shown in Table 2), so it should be used as the baseline. The results demonstrate that indeed, savings are greatest if scenario 8 occurs, but that there are also LCC savings on average across all scenarios.

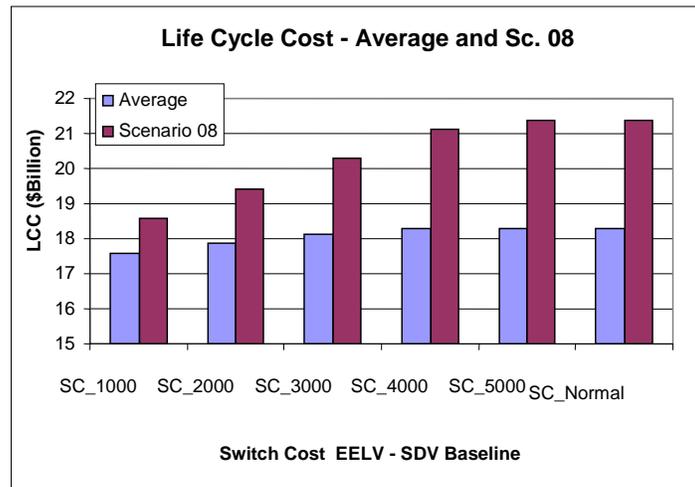


Figure 13: Total Life Cycle cost as a function of switching cost between EELV (3) and SDV vehicle (1)

Figure 14 makes this even more apparent. It shows the exact amount saved as switching costs are lowered between the two most-used vehicles.

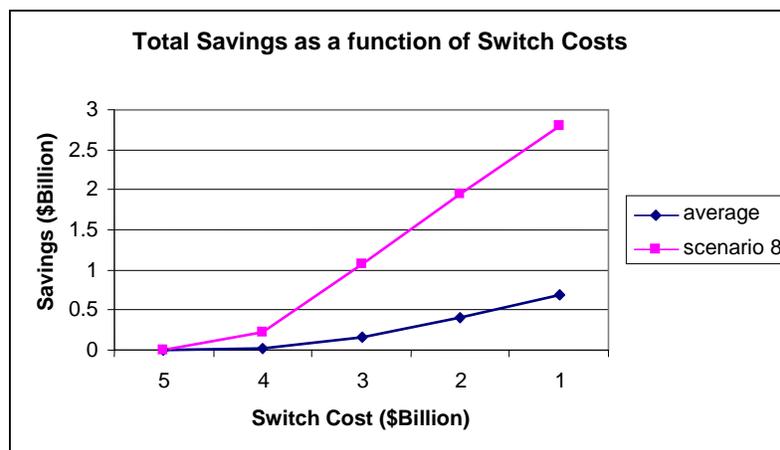


Figure 14: Total LCC Savings as a function of switching cost

If we think all scenarios are equally likely, we can save up to \$600 million by lowering the switching costs to \$1 billion dollars. If we think that scenario 8 will probably occur, we can save as much as \$2.8 Billion by lowering the switching cost between configurations 3 and 1 to \$1 billion. This, therefore tells us exactly how much we should be

willing to pay to lower the switching costs to those desired numbers. This in a way is the maximum price we would be willing to pay on a real option that allows (but doesn't require) switching from configuration 3 (EELV) to 1 (SDV) at a later date. If it is cheaper to lower the switching costs than not to switch and incur future costs due to inefficiency, then the switching costs should be lowered. If this is not the case, then it makes more sense to keep the switching costs where they are. Two final points should be noted. First, we can weight scenarios as we please rather than resorting to averages. Second, lowering the switching cost might change the performance characteristics of the system, and therefore change the life-cycle costs. This should be taken into account during iterative design.

IV. Conclusions

TDN represents an advancement over similar methods to design and analyze system flexibility. First it is formulated to enable designers to explicitly address the problem of high switching costs, a principal cause of architectural lock-in, early in the design process in a clear and rigorous manner. Second, by exploiting the inherent structure of the time-expanded graph, TDN overcomes the problem of exponentially branching development options, a serious impediment to the practical implementation of both Real Options and Decision Theory-based design methods. In this way, TDN reformulates life-cycle evaluation as a network optimization problem with clearly defined user inputs. Third, TDN enables quantitative, user-directed scenario planning. Fourth, TDN lends itself easily to implementation as an iterative computational tool, reducing the cognitive load on system designers and allowing them to concentrate on design alternatives, switching options and development scenarios. Finally, the method is generic, being applicable to any systems and sub-systems where architectural lock-in and the high-switching costs are a problem. Examples of applications that could use TDN would be the design of a factory under uncertain capacity requirements, or fleet acquisition for an airline under uncertain traffic models.

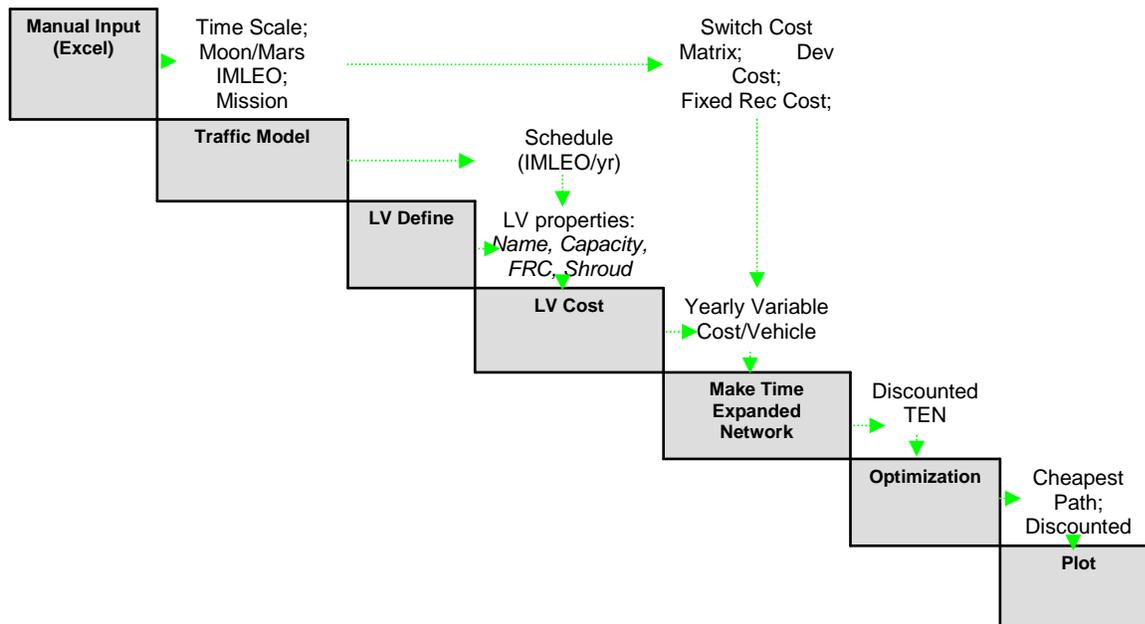
Some notes on the TDN Model Formulation

The generic modeling framework described above can be implemented in many different ways based on specific modeling demands. Some issues raised in this paper are briefly noted below.

- *Complete Information:* Classic Shortest Path Algorithms assume perfect information about future operating costs. In reality operators do not know what operating scenarios will exist in the future. This can be taken into account by revising the shortest path algorithms and only taking into account fewer future time periods. However, one must be careful in implementing this, as the goal for the model is to increase system flexibility and not to recreate an operating environment per se. It is likely what is needed early on is an assumption of complete information together with the most likely demand scenarios, to identify how to maximize useful switches.
- *Number of Time Periods:* We can split the life-cycle into an arbitrary number of time units with each time unit representing an arbitrary amount of real-world time.
- *Time to switch:* Switching time can be modeled in similar way to classic "Project Cost Curve" problems. In these it is generally assumed that a project is an ordered set of jobs. Each job has an associated normal completion time and a crash completion time, and the cost of doing the job (in our case, making the switch) varies linearly between these two times. Project cost curves also have a long history (*Ford & Fulkerson p. 151*).
- *Mixed Strategies:* We may want to operate more than one kind of a system at time. This can be done by sending more than one unit of flow through the network. This is an area for future study.

TDN developed from the realization that, if properly framed, switching costs between system configurations define a network whose nodes are point-designs and arc-costs are both operating and switching costs. Using concepts from operations research, this "static network" can be expanded into a "time expanded network" in order to model a complete life-cycle. The time expanded network provides a compact representation of all possible development paths for the set of point designs, and thus avoids exponentially branching development options often encountered in traditional decision theory. Used in conjunction with established cost modeling techniques including the estimation of fixed and variable operating costs, the time-expanded network model can be implemented as a software tool and used iteratively to identify optimal switches and design increasingly complex evolvable systems.

Appendix : N-Squared Diagram of the Software Tool Created to run TDN



References

- ¹ Ashby, William Ross. *Design for a Brain: The Origin of Adaptive Behavior*. Science Paperbacks. London, 1966
- ² Cowan, Robin. "Nuclear Power Reactors: A Study in Technological Lock-In." *Journal of Economic History*, Vol. 50, No. 3 (Sep., 1990), pp. 541-567
- ³ Christensen, Clayton M. *The Innovator's Dilemma*. Haper Collins Publishers Inc., New York, NY. 1997
- ⁴ Henderson, Rebecca M. et al. "Architectural Innovation: The Reconfiguration of Existing Systems and the Failure of Established Firms." *Administrative Science Quarterly* (35), 1990, p. 9-30
- ⁵ Liebowitz, SJ, et al. "Path Dependence, Lock-In, and History." *Journal of Law, Economics, and Organization*, vol. 1 no. 1. 1985
- ⁶ Katz, Michael. "Systems Competition and Network Effects." *Journal of Economic Perspectives*. Vol 8, no 2, 1984. p 93-115
- ⁷ Witt, Ulrich. "Lock-In v Industrial Mass; Industrial Change Under Network Externalities" *International Journal of Industrial Organization*. 15, 1997 p 753-ff.
- ⁸ Utterback, James M. "Innovation in Industry and the Diffusion of Technology." *Science*, New Series, Vol. 183, No 4125 (Feb. 15 1974), p. 620-626
- ⁹ Sarsfield, Liam "The Technology Puzzle: Quantitative Methods for Developing Advanced Aerospace Technology". RAND, National Security Research Division, 2001.
- ¹⁰ Puffert, Douglas J. "Path Dependence, Network Form, and Technological Change" Article in: *History Matters: Economic Growth, Technology, and Population*. Stanford University Press, November 1, 2003
- ¹¹ Dixit, Avinash. "Investment and Hysteresis." *Journal of Economic Perspectives*. Volume 6, No 1. Winter 1992. P 107, 1932
- ¹² 74. de Weck O.L., Suh E.S., "Flexible Product Platforms: Framework and Case Study", DETC2006-99163, Proceedings of IDETC/CIE 2006 ASME 2006 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, September 10-13, 2006, Philadelphia, Pennsylvania USA
- ¹³ Haberfellner R., de Weck O.L., "Agile SYSTEMS ENGINEERING versus engineering AGILE SYSTEMS", *INCOSE 2005 - Systems Engineering Symposium*, Rochester, NY, July 10-15, 2005
- ¹⁴ Simon, Herbert A. *The Sciences of the Artificial*; Third Edition. MIT Press. Cambridge MA 1996
- ¹⁵ Wertz, James and Larson, Wiley. "Space Mission Analysis and Design, Third Edition." Jointly published by Microcosm Press, El Segundo, CA and Kluwer Academic Publishers, London. 1999 (P 786)
- ¹⁶ de Weck, O., 2005, "Determining Product Platform Extent", *Product Platform and Product Family Design: Methods and Applications*, T. W. Simpson, Z. Siddique and J. Jiao, eds., Springer, New York, pp. 241-301
- ¹⁷ Ford, and Fulkerson. *Flows in Networks*, Princeton University Press. Princeton, NJ. 1962
- ¹⁸ Ahuja R.K., Magnanti T.L., Orlin J.B., *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, February 1993
- ¹⁹ de Weck, O.L., de Neufville R. and Chaize M., "Staged Deployment of Communications Satellite Constellations in Low Earth Orbit", *Journal of Aerospace Computing, Information, and Communication*, 1, 119-136, March 2004
- ²⁰ Silver, M., "Designing Sustainable Heavy Lift Launch Vehicle Architectures: Adaptability, Lock-In and System Evolution", S.M. Thesis, Department of Aeronautics & Astronautics, MIT, September 2005