Discrete Event Simulation of Reusable Launch Vehicle Ground Operations (RLVSim)

AE8900 MS Special Problems Report
Space Systems Design Lab (SSDL)
School of Aerospace Engineering
Georgia Institute of Technology
Atlanta, GA

Author
John Daniel Reeves Jr.

Advisor
Dr. John R. Olds
Space Systems Design Lab (SSDL)

July 30, 2004

Table of Contents

List of Figures

List of Tables

Acronyms

AATe	Architecture Assessment Tool – enhanced
BPF	Booster Processing Facility
CMRG	Combined Multiple Recursive Generator
CPD	Cost per Day
DDT&E	Design, Development, Testing & Evaluation
DES	Discrete Event Simulation
ET	External Tank
FEL	Future Event List
FIFO	First-In, First-Out
ISS	International Space Station
LCC	Life-Cycle Costs
LOV	Loss of Vehicle
MMH	Monomethylhydrazine
NASA	National Aeronautics and Space Administration
OMB	Office of Management and Budget
OMS	Orbital Maneuvering System
OPF	Orbiter Processing Facility
PDF	Probability Density Function
RCS	Reaction Control System
RLV	Reusable Launch Vehicle
RMAT	Reliability and Maintainability Analysis and Estimation Tool
RNG	Random-Number Generator
SCAPE	Self-Contained Atmospheric Protective Ensemble
SRB	Solid Rocket Booster
SSDL	Space Systems Design Lab (Georgia Tech)
STS	Space Transportation System
TBCC	Turbine-Based Combined-Cycle
TPS	Thermal Protection System
TSTO	Two Stage to Orbit
VBA	Visual Basic for Applications

1.0 Introduction

One of the most important aspects of any major transportation vehicle, whether it be a commercial aircraft, reusable launch vehicle (RLV), or any other highly complex design, is the cost and time associated with readying it for use. Once a design of this sort has been developed, tested, and declared operational, the life-cycle costs (LCC) become the dominant metrics of interest and are primarily made up of the costs and times associated with scheduled maintenance, unscheduled maintenance, part replacement, and refurbishment. These parameters are usually encapsulated in the "operations" discipline, which focuses on the operational concepts required to maintain complex vehicles.

Although operations analyses are essential to any complex vehicle, access-to-space vehicles require more attention in this area than any other type of vehicle since their safety and performance margins are usually very slim. Re-entry vehicles go through rigorous thermal environments that require that highly sophisticated thermal protection systems (TPS) be incorporated into the design. TPS maintenance is always a high operational driver since the materials used, whether they be tile-like or ablative, have to be carefully inspected prior to every mission. Because access-to-space vehicles have to operate in the demanding environment of space for extended periods of time, they require a high number of sophisticated subsystems to ensure that any payload or crew are accommodated and safely delivered or returned. A higher subsystem count usually translates into a higher processing load when the vehicle is on the ground. Because of this, a large work force is generally required in order to provide the man-hours for the intricate inspections and refurbishment. Large workforces result in very high labor costs, which is usually one of the biggest drivers in vehicle's operational cost. Because of such consequences of operating complex vehicles, it is imperative that these considerations are taken into account as early as possible in the design phase.

A lot of time and money has been invested in technology development over the years to alleviate the ground operations environment of space access vehicles, with RLVs in particular being the main focus since the Space Transportation System (STS) has been the dominant design over the past few decades. A highly reusable space access vehicle with low turnaround requirements would open up an entirely new paradigm in the space industry since it would lead to routine access to various orbital assets as well as allow the space tourism market to truly unfold. However, operational considerations are still sometimes ignored during the conceptual design phase in lieu of focusing attention on the design's general performance and capability. A proper design takes operational cost considerations into account as a secondary objective behind performance requirements, which can lead to a design that meets mission objectives such as payload performance but does not require

heavy operational burdens. However, designers often choose to invest less money up front, usually due to funding constraints, instead of investing heavily in new technologies early on that can reduce operational complexity. The result is that the design is operationally more expensive, and sometimes less reliable, than envisioned once the first units begin to enter service. A prime example of such a scenario is the STS itself. Despite the successes of the Apollo program in the late 1960s and early 1970s, the Nixon administration began to downsize NASA's budget and focus the spending on other non-space related programs during the mid 1970s. Grand goals of building a space shuttle along with an enormous space station and other orbital assets were quickly forgotten as the budget was reduced. NASA designers were eventually allowed to develop just the Space Shuttle system under a constricting $5.5 billion ceiling[1] enforced by the Office of Management and Budget (OMB). STS was designed and developed into a working system, but many operationally costly decisions were made during the conceptual design phase in order to keep design, development, testing & evaluation (DDT&E) costs as low as possible. For example, the high number of subsystems that were incorporated into STS during its design phase led to a vehicle that is very difficult to refurbish and maintain. Due to subsystem complexity, the Space Shuttle Main Engines (SSMEs) have to be completely removed from the orbiters between each flight and processed separately, which directly leads to longer turnaround times. Another example is the TPS tiles used to protect the orbiter during re-entry. Almost every single tile used on the orbiters is unique and has to be individually inspected between each flight. Because of these types of design decisions, STS currently costs around $300 million dollars[2] to launch, which is drastically higher than the $7.7 million per flight[2] that was being predicted by NASA in the early 1970s. This high launch cost is due to the fact that STS launches only six or seven times a year at best – such a rate being significantly less than the 50 launches per year[2] that NASA originally predicted. From this example, it is apparent that operational considerations can make a tremendous impact on a concept's LCC, and so should always be one of the primary factors during the conceptual design phase. By bringing operational knowledge and analysis forward (earlier) in the design phases, a much more efficient system can be developed.

Since the early design phases of access-to-space vehicles take place on a conceptual level, it is usually difficult to forecast exactly what the concept of operations will consist of when such systems are operational. For example, the type of general propulsion propellant is usually decided upon fairly early in the design process, but smaller propulsion elements such as reaction control systems (RCSs) and orbital maneuvering systems (OMSs) are not specifically designed until later design phases. These elements can have significant impacts since they could possibly require the loading and unloading of hazardous materials/propellants. NASA currently requires that all operations cease during STS

processing while personal wearing SCAPE suits purge, drain, and load materials such as the MMH used for OMS and RCS. This leads to a significant delay in operations turnaround that could have been avoided had the original design used alternate, safer propellants.

Many tools have been developed to assist in analyzing the operational demands on a vehicle during the conceptual design stage. Two commonly used examples of these types of tools are NASA Langley's Reliability and Maintainability Analysis and Estimation Tool[3] (RMAT) and NASA Kennedy's Architectural Assessment Tool – enhanced[4] (AATe) that both parametrically estimate operational parameters based on user inputs. RMAT requires the user to input various subsystem and top-level system parameters in order to generate estimates for component reliability and scheduled and unscheduled maintenance requirements. Likewise AATe requires inputs such as vehicle scaling information, engine count, processing flow paths, and so forth. RMAT is useful in generating detailed maintenance information that can be used to plan facility support levels as well as identify which subsystems are the most operationally problematic. AATe generates the overall fixed and variable operations costs as well as the general turnaround time per vehicle. AATe is based on extrapolated STS historical data, while RMAT uses STS data in conjunction with numerous military aircraft operations data in order to allow the user to design/analyze vehicles that fall in the middle of the design space bounded by traditional access-to-space vehicles (STS) and typical military aircraft. Both of these tools are commonly used in industry to obtain deterministic estimates of operational characteristics of access-to-space vehicles.

Although the aforementioned tools are useful for conceptually estimating various operational demands, a new form of analysis will be commonly required for future RLV studies. Discrete Event Simulation (DES) is a methodology that has evolved rapidly over the past few years and can be used to dynamically study the operational flow paths of vehicle operations. Various probabilistic decision points can be introduced into models that lead to more accurate modeling of real world circumstances such as LOV scenarios. DES models can also provide confidence intervals based on data samplings from multiple replications that offer insight into how accurate the estimated values are. Models have already begun to be developed that focus on the turnaround operations of access-to-space vehicles. One example of such a tool is GEM-FLO[5] (Generic Simulation Environment for Modeling Future Launch Operations) that was developed by Productivity Apex Inc. in conjunction with NASA Kennedy. GEM-FLO uses a graphical user interface that allows users that are not familiar with DES to obtain DES-generated results focusing on ground turnaround operations (flight rate, facility utilization, etc.). This model is powerful in that it takes a generic approach to vehicle processing modeling by assuming that almost all access-to-space vehicles have commonality amongst the types of phases that are required for launch.

This report will detail the development of a similar DES model tool (RLVSim) that models ground turnaround operations specifically for reusable launch vehicles. The main benefit from RLVSim is that it is tailored for use with Arena's educational mode so that it can be used by without a professional version of Arena, allowing students and student researchers a chance to utilize DES in vehicle design studies. The tool is also tailored to be used in conjunction with specific other design tools, specifically AATe, that are commonly used in space vehicle design. It should be noted that RLVSim makes the assumption that there is an unlimited market demand for payload delivery, meaning that it forecasts the maximum achievable throughput based on the input parameters. The costs that are accumulated in the model are also tallied in a steady-state fashion, meaning that increasing or decreasing trends do not occur throughout the simulated years in response to an aging fleet or a processing learning curve. However, RLVSim allows various trade studies and concept of operations analyses to be made in addition to providing probabilistic ground operation estimates that will take into account real world uncertainties. Distributions associated with the various turnaround facilities mimic the variability associated with such work, and a LOV probability is incorporated for each vehicle being analyzed that has an impact on the overall turnaround statistics. This report will document such details pertaining to the development of RLVSim, as well as provide example studies that demonstrate how the tool can be used to gain insight on RLV LCC aspects.

2.0 Discrete Event Simulation

Discrete Event Simulation, like many other computer-aided tools used today, has seen a rapid evolution in the past few decades due to increased computational power and advances in DES package capabilities. This section will outline the background and history of DES as well as provide a discussion of the particular software used for this project. It will conclude by discussing basic simulation theory along with explaining what takes place when a DES model is run.

2.1 DES Background

Discrete event simulation is a numerical computer-based simulation technique that has been under development for the past few decades. General computer-based simulation has roots that trace back to the 1950s when computer programming became popular, but not until the past couple of decades has DES become a viable technique. DES' most vital benefit is that it combines the relative ease and flexibility of computer programming with the crucial results of statistical analysis. The late 1970s and early 1980s saw the emergence of numerous DES simulation languages such as GPSS, SIMSCRIPT, SLAM, and SIMAN[7] that began to capture the effects of combining statistics with computer programming. These languages became industry standards due to their applicability to almost any sort of engineering, manufacturing, or any other queue-based field. DES quickly became associated with the field of Industrial Engineering due to its inherent statistics foundation, as well as its popular application to manufacturing, transportation, and other logistics-based activities. The early simulation languages such as SLAM, SLAM II, and SIMAN were powerful, but required a heavy amount of programming and a significant learning curve. Due to the boom in computing power of the early 1990s along with an increase in graphical user interfaces (GUIs) popularity, advanced simulation products that combined the power of the early simulation languages such as SIMAN with the ease of use and reduced complexity of a graphical environment began to hit the market. In addition to having a graphical coding interface that reduced the learning curve required, some packages began to include sophisticated animation capabilities that not only assisted users in troubleshooting, but also gave users a way of demonstrating model logic and dynamics to others.

2.2 Rockwell Software's Arena

Rockwell Software's Arena[6] is a powerful package that has consistently been a top selling product of the DES software industry for the past several years. Its popularity can be traced to its ability to provide useful results without requiring too significant a learning curve. One of Arena's most beneficial traits is that users across the whole spectrum of skill-

levels can use the product to generate useful results. This robustness is achieved by expanding upon an evolved version of the SIMAN language, meaning that Arena has been built upon the shoulders of an already successful product. Arena allows users to choose from various *modules* that are presented in various *templates* ranging from basic logic pieces to complex items such as conveyers and transporters. Each module represents a combination of SIMAN code that has been pre-packaged to allow the user to drag and drop pieces of code into the model without having to work with the code itself. In fact, an entry-level user can design, develop, and execute somewhat complex Arena models without having to type a single line of code. Arena also provides generated reports at the end of simulation runs that can be modified however the user sees fit.

Despite being straightforward enough for a beginner user to use, Arena allows experienced users to model at sophisticated levels of detail. Each of the modules is basically a combination of various pieces of SIMAN code that have been packaged together for the more popular coding scenarios. Arena also provides a *blocks* template that contains the individual pieces of logic that make up the pre-packaged modules. For example, a *process* module in the basic process template contains logic to seize and release a *resource* along with logic to delay the process for a specified duration in the interim. In the *blocks* template a user can find each of these logic pieces, such as *seize*, as individual pieces that can be added to the model. This allows a user to combine any of these logic pieces as they see fit in order to achieve the modeling logic needed.

In addition to the basic SIMAN blocks that can be used to write the model logic at a basic level, Arena also allows users to include pieces of code in other languages such as Microsoft's Visual Basic for Applications (VBA) or C. For example, every time an *entity* passes through a *VBA* module, a corresponding piece of VBA code can be executed. This is very useful in that it allows the use of ActiveX® object libraries common in most PC desktop applications so that Arena can interact with other programs and vice versa. Arena does provide read and write modules in the advanced template that allow models to read and write to Excel, Access, or regular text files, but the addition of embedded VBA code allows an unlimited amount of communication between applications. This in turn can lead to the implementation of Arena into engineering software design suites such as Phoenix Integration's ModelCenter®.

2.3 Arena Details

Arena's main interface is a working space common among Windows-based environments along with a template window that allows users to drag and drop modules onto the working space. Table 1 contains generalized definitions to the most common simulation terms encountered. A *model* is basically the simulation scenario itself that

encapsulates everything going on in the simulation. An *entity* however is an actual dynamic piece that proceeds through the model while interacting with various *processes*. Many processes require the access of *resources*, and can be used to model just about any real world activity. The *resources* themselves are whatever an entity may need to interact with during a process. *Variables* and *expressions* are model-specific parameters that are independent with any entity or resource, although they can be accessed for information anywhere in the model. Again these are just generalized definitions because it is up to the user to model various real-world events appropriately. For instance strap-on solid propellant stages such as the shuttle's SRBs could be modeled as entities or resources. They are entity-like in that they move from one place to another and access different resources (such as assembly facilities), but they are also resource-like in that they are seized by the shuttle itself during the integration, pad preparation, pad, and launch phases. It is the modeler's responsibility to appropriately model such situations.

Table 1. Basic Arena Definitions.

Term	Definition
Model	A combination of processes and process flows that represents a real-world scenario. A typical model would include many different aspects of various scenarios such as queues that correspond to various processes along with the various entities that travel throughout the system.
Entity	The fundamental driver of a simulation that represents what is using or accessing the various processes. Entities travel throughout the simulation model and are generally the dynamic pieces of the model that change throughout time.
Process	A capture-all term that is used to define various stops along an entity's path that require the interaction with resources. Processes are used to model activities such as interacting with a bank teller or using a launch pad.
Resource	A resource is any external service or item that an entity needs to interact with during a process. For example, if an entity goes through the process of interacting with a bank teller, the actual teller is a resource. A particular manufacturing machine that is accessed via a process is itself a resource. A resource is seized and released as needed.
Attribute	Attributes are pieces of information that are related to the various entities. A simple example would be if balls were modeled as being either a red ball or a green ball. One of the ball's attributes would be the color.
Variable	A variable is a model wide parameter that is not related to any one particular entity or process. Variables can be updated through a simulation run as needed.
Expression	An expression is similar to a variable in that it does not pertain to any one entity or process, but differs in that it is generally used to model mathematical relationships or statistical expressions.

There are many modules that have been included in the basic and advanced process templates, many of which are used on a constant basis. The *Assign* module is perhaps the most commonly used of all, for it allows for manipulation of any attribute or variable. For example, a cost attribute associated with each entity could be updated via an *Assign* module every time an entity passes through. The *Batch* and *Separate* modules allow users to combine and separate various entities either temporarily or permanently while specifying various attributes of the entities to be included in the resulting entity batch. The *Decide* module can be used as a decision point based on either a probability metric or a particular attribute, variable, or expression value. The *Record* Module allows a particular metric to be updated and reported at the end of the simulation run. For example, a *Record* module could be used to tally a cost value associated with each entity once they have made it through a particular process. The report automatically generated at the end of each simulation run can average tally statistics recorded using the *Record* module. The *Route* module can be used for two different purposes. First, it allows entities to transfer from one place in the logic stream to another without having the modules directly linked, which can be used to better organize the various modules. Second, *Route* modules can be used as animation markers for animated simulation runs since they designate where various entities are sent in the model logic.

Table 2. Commonly Used Arena Modules.

Module Name	Description
Create	Used to create entities
Dispose	Used to dispose entities
Process	Used to seize and release resources as well as delay entities as needed
Decide	Used as a decision point in model logic based on probabilistic choice or attribute or variable condition
Batch	Used to combine entities into a single entity
Separate	Used to separate a batched entity into its constituent pieces
Assign	Used to manipulate attributes or variables every time an entity passes through
Record	Used to record tallies or counters that can be used to average statistics at the end of the simulation
Delay	Used to delay entities during logic flow
Hold	Used to hold an entity in a queue until a specified condition is met
ReadWrite	Used to read or write from an external file (i.e. Excel, Access, or text file)
Release	Used to release a seized resource
Seize	Used to seize a resource
Route	Used to route entities from one section of the model logic to another

Table 2 contains a short description on many of the aforementioned commonly used Arena modules. Listed are the modules used by the RLVSim model, in conjunction with several

Blocks template modules, to simulate RLV turnaround vehicle processes appropriately. Arena also includes several other templates such as "Factory Elements" and "Packaging" that contain modules that are specific to logistic or manufacture industries.

An Arena model is executed by using the "run" or "fast-forward" buttons located in the top toolbar. Simulation speed can be increased or decreased as needed using buttons on the toolbar or by using the "<" or ">" keys. The animation speed factor is displayed at the bottom left-hand corner of the screen while adjustments are being made, and can range from 0.000010 to 100. If the speed factor is set to a low enough speed, the various entities can be seen progressing through the model logic. If animation stations have been set up in a separate animation area, the entity pictures can be seen moving from station to station. This technique can be used to develop elaborate animations to demonstrate the model logic. If the main purpose of a model is to generate model statistics, then animation should probably be kept to a minimum since it does require extra time to run. Also, the run setup option under "run" can be used to modify the simulation parameters such as number of replications, the base time units to be used, replication length, warm-up period length, or report generation options. After a model has been run, Arena waits for the "stop" command to be given before resetting the model.

Arena version 7.01 was utilized for the purposes of this project in conjunction with Windows XP professional and a 2.8 GHz. Desktop PC with 512 MB of RAM. The model was developed using Arena's educational mode in order to allow any user to work with the model without having to acquire a professional license. The textbook *Simulation with Arena* by Kelton, Sadowski, and Sadowski (2004) contains a CD-ROM copy of educational version of Arena, which allows any user to develop models, but with a ceiling on the number of modules or SIMAN objects (attributes, variables, modules, etc.) that can be introduced to a model. Arena version 7.01 was packaged with the third edition of the book, while the upcoming edition will be packaged with Arena version 8.00[7]. Version 8.0 will include improvements such as ActiveX® controls that allow dynamic interaction with simulation runs, rotating symbols that allow for improved resource animation, increased macro-recording capabilities, in addition to several other improvements[8].

2.4 Discrete Event Simulation Theory

Simulation theory is made up of several statistical techniques such as queuing theory. The essence of DES according to a popular DES text is that it is "the modeling over time of a system all of whose state changes occur at discrete points in time"[9]. State changes are events that are considered changes in the model such as any time an entity is transferred from one module to another, any time an attribute/variable is changed, or any time a resource is seized or released. This is demonstrated by an Arena model's clock as a

simulation progresses. Instead of changing from day 1 to day 2 to day 3 and so forth, assuming the model's base time unit is set to days, the clock will jump from day 3 to day 52.5 or some other time based on what is happening in the system. This scheduling mechanism is accomplished by using what can be called a "future event list", or FEL[9], or an "event calendar"[6]. An FEL essentially keeps track of what is currently going on in the model during each step. For example, if an entity enters into a *process* module and is delayed while seizing a resource, the FEL will determine and keep track of how long that entity will be tied up in the delay by predetermining the delay end date as soon as the delay starts. This determination is achieved by pulling a random value from a specified distribution. Thus this technique is called "Discrete Event" simulation, for everything that happens in the model does so on a discrete step-by-step fashion. If two events are scheduled to happen at the exact same time, they are executed serially based on when they entered in the FEL. This method can be used to adequately model just about any real-world scenario, for even any event that takes place in a continuous fashion such as fuel usage, object displacement, or power output, can generally be modeled discretely.

Much of DES is built upon queuing theory since entities spend almost all of their time either using a resource or waiting in a queue of some sort. Arena's queues usually operate in a first-in, first-out (FIFO) fashion, meaning that entities are modeled like cars at a red-light. The first car to the light is the first one that proceeds through when the light turns green. In an Arena model, entities arrive at a resource and wait in the queue until that resource is open for use. The entity that arrives at the resource first gets access to the resource before others. The exceptions to this rule are when resources are seized based on priorities or when entities of a certain type are batched out of a batching queue based on some attribute. The time that is spent in various queues can be summarized by the following equation[6]:

$$\textit{Average time in queue} = \frac{\sum_{i=1}^{N} WQ_i}{N} \tag{1}$$

where WQ_i is the i[th] entity's waiting time in a particular queue, and N is the number of entities that enter the queue during a simulation run. The automatically generated Arena report contains average waiting times for all queues in the model by using this equation. It is useful in that it provides a means of identifying where entities are spending their time in the system. The report also provides a minimum and maximum waiting time experienced by entities.

Another beneficial queuing theory calculation[6] that is used by the report generator is the time-average number of parts waiting in the particular queues:

$$\textit{Time-average number waiting in queue} = \frac{\int_0^L Q(t)dt}{L} \tag{2}$$

where $Q(t)$ is the number of parts in the particular queue at time t and L is the simulation length in base time units. This statistic adds visibility to where the bottlenecks are occurring in any particular simulation, as well as what kind of facility space needs to be allocated to the resource. For example if a simulation of banking transactions indicates that the time-average number of people waiting for a teller is 10 people, then adequate space for at least 10 people should be made in the waiting lines. The automatically generated report also includes the minimum and maximum values for these statistics.

Queuing theory, while simple in nature, can provide powerful insight into what is happening in a particular system or model. The most problematic occurrence in any model is extensive queue times, since these drive overall throughput and usually capture a majority of an entity's life.

Resource utilization is another area that is vital to simulation studies. If entities are not waiting in some sort of queue, then they are probably busy interacting with a resource (assuming transfer times are minimal). Resources are assigned a capacity either on a fixed-capacity basis or by schedule. A resource's capacity is a direct modeling translation to how many entities with which the modeled machine/person/etc. can interact. The capacity of a teller resource will usually be one, while an automotive repair shop may be able to dual-process two vehicles at the same time. Long queue lines can usually be related to very high utilization metrics for queues, while very low utilization numbers can be seen as identifiers for overly-equipped resources. If a resource's capacity is representative of the number of employees on-shift, then a low utilization number may indicate that the scheduled shift is over-manned. The basic resource utilization metric can be calculated as follows[6]:

$$\textit{Average resource utilization} = \frac{\int_0^L B(t)dt}{L} \tag{3}$$

where $B(t)$ is equal to 1 if the resource is busy or 0 if the resource is idle at time t and L is the simulation length in base time units. The Arena report provides this metric along with the maximum and minimum utilization values for each resource in a particular Arena model.

The *Record* module in Arena is commonly used to record user-specified metrics in the simulation report based on entity-related attributes such as an arrival time, or in some cases a replication specific value such as a total cost. Every time an entity passes through the *Record*

module, the specified metric is recorded and averaged by the numbers of entities that have passed through. This holds true whether the simulation is replicated one or many times. The report documents the average, minimum, minimum average, maximum, and maximum average value of all of the user-specified metrics.

The Arena simulation report also contains "half-widths" for every metric that is presented. These half-widths are actually 95% confidence interval half-widths that were calculated using the following[6]:

$$95\%\ confidence\ interval\ half\text{-}width = t_{n-1,1-\alpha/2}\frac{s}{\sqrt{n}} \tag{4}$$

where $t_{n-1,\alpha/2}$ is the upper 1- $\alpha/2$ t-statistic critical point with n-1 degrees of freedom, s is the sample standard deviation, and n is the number of samples. Sometimes the report will present a half-width as "insufficient", which means that there were not enough data points for the equation to be accurate. A 95% confidence interval is a normal distribution related concept, so there is an inherent assumption that the data points are normally distributed.

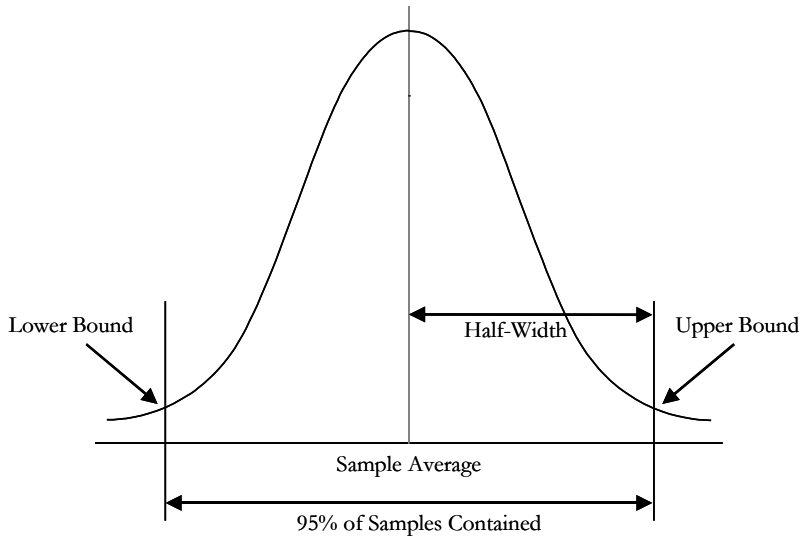


Figure 1. 95% Confidence Interval Half-Width on a Normal Distribution.

The Central Limit Theorem in statistics[10] stages there has to be a certain minimum of data points in order for a normally distributed trend to appear in data. If there are not enough data points for this to happen, then Arena declares the half-width fields as "insufficient".

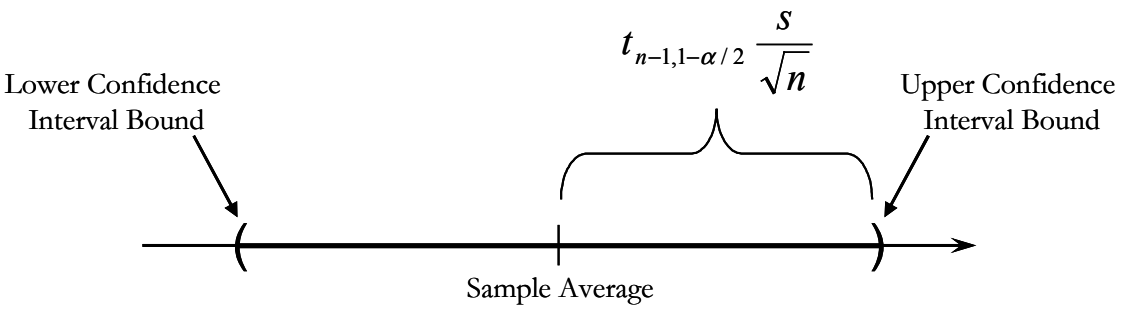


Figure 2. 95% Confidence Interval Half-Width.

Arena allows any time-related event to be distributed in just about any type of distribution the user sees fit. The major choices presented to the user are beta, erlang, exponential, gamma, johnson, lognormal, normal, poisson, triangular, uniform, and weibull. If a user wishes to gather random data from a distribution not mentioned above, then the user can code in the probability density function (PDF) of that distribution. The three major distributions used in RLVSim model are the exponential, normal, and triangular distributions. Only the arrival times of the initial set of orbiters and booster stages are modeled as exponential, which only happens a few times in each simulation run, so this distribution is not overly important. Exponential is Arena's default distribution for such cases since it only requires a nominal mean value. The normal distribution is used only in the sense that the output data points have to behave normally distributed in order for the aforementioned half-widths to be calculated. The triangular distribution is what is used to model all time distributions, and is a good representation of real-world occurrences since the physics of the problem usually dictate a minimum value that can be achieved. Figure 3 contains the PDF for a triangular distribution. The basic parameters are the minimum value (*a*), the most likely value or mode (*m*), and the maximum value (*b*).

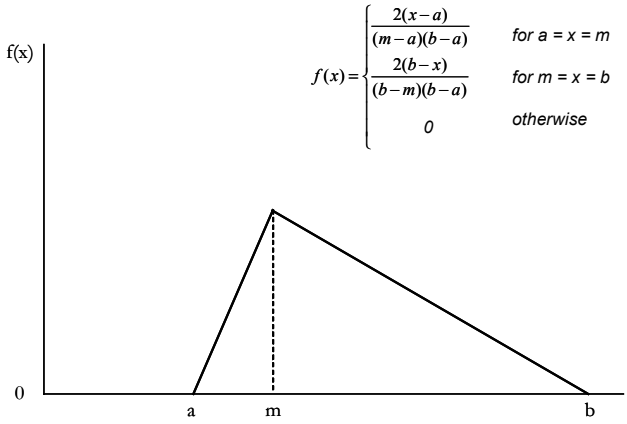


Figure 3. Triangular Distribution Probability Density Function.

The triangular distribution is useful in that it bounds the range of possible values, and can be used to weigh the randomization process towards either bound.

Although discrete event modeling uses a heavy amount of stochastic data generation, any computer generated random number is not truly "random" in the sense that it is not completely a spontaneous occurrence. Computers actually rely on *pseudorandom*-number generators that use complex algorithms that create the appearance of random numbers, but provide quite repeatable results. In fact, these numbers will pass any sort of statistical distribution test and are no different than a true random number for most intents or purposes. The benefit from this is that the generations are repeatable, allowing for easier debugging. If an Arena model is set to only replicate once, and no variables or any other model specific parameters are changed, every run will result in the same results. A constant random number seed is used during every run that causes the same "random" numbers to reoccur. When a simulation is replicated, a different seed stream is chosen each time, which creates the variability that is essential for the report calculations. Recent versions of Arena use a complex random number generator (RNG) called a combined multiple recursive generator[6] (CMRG) which uses the following:

$$A_n = (1403580 A_{n-2} - 810728 A_{n-3}) \bmod (4294967087) \tag{5a}$$

$$B_n = (527612 B_{n-1} - 1370589 B_{n-3}) \bmod (4294944443) \tag{5b}$$

$$Z_n = (A_n - B_n) \bmod 4294967087 \tag{5c}$$

$$U_n = \begin{cases} Z_n / 4294967088, Z_n > 0 \\ 4294967087 / 4294967088, Z_n = 0 \end{cases} \tag{5d}$$

Where A and B are a six-vector stream of seeds and U_n is a random number between 0 and 1. The CMRG is a recently developed technique that generates a random number stream cycle that has a length of $3.1 * 10^{57}$. Arena further breaks this stream down into substreams that are used uniquely during each replication of a model.

3.0 RLVSim Arena Model

The RLVSim Arena model uses a variety of Basic Process, Advanced Process, Advanced Transfer, and Blocks modules to model the turnaround operations of typical single or two-stage reusable launch vehicles. Many vehicle and model specific parameters can be changed to parametrically study the output metrics. These metrics consist of average vehicle turnaround, average vehicle recurring cost per flight, total recurring cost per simulation, total launches per replication, average launches per year, and the average loss of vehicle (LOV) rate. The model can be used in either of three different fashions: single replication run (for use with multidiscipline design suites such as Phoenix Integration's ModelCenter® and debugging), a Monte Carlo run (multiple replications), or a single run with Excel output (to view processing schedules).

3.1 RLVSim Arena Model Logic

The RLVSim model consists of logic modules that provide a high-level modeling of RLV turnaround operations as well as various mechanisms used to record the cost and turnaround metrics of interest. The different entities that represent either an orbiter or a booster proceed through the processing, integration, runway/pad, ascent, and depot maintenance phases of the operations.

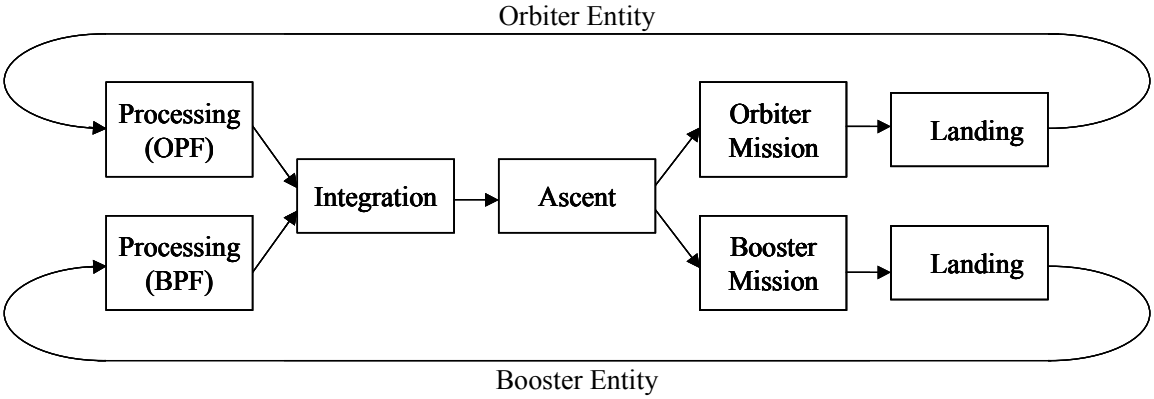


Figure 4. Top-Level Process Flow.

The three major phases all have a cost per day factor associated with them that is used to determine the variable cost per flight based on the number of days the orbiters spend in each of the facilities. Each time an orbiter or booster entity arrives at a particular

facility, Arena draws a random value from the assigned triangular distributions and places that entity in a delay for that specified amount of time. If an orbiter or booster arrive at a facility resource that is already in use (utilization = 1), then they wait in a queue until that resource becomes available.

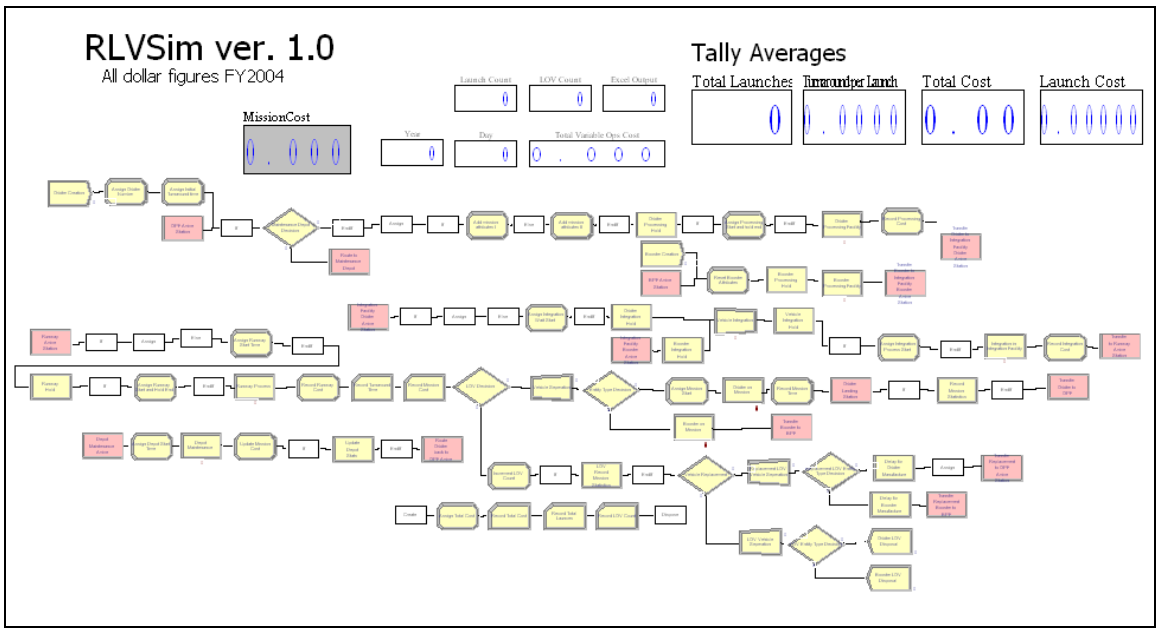


Figure 5. RLVSim Screen Capture – Entire Model.

Because the waiting time in the queues can become a significant portion of the orbiter's time spent in the process flow, the user has two options regarding how to treat the variable cost while in queue. The first is straightforward in that waiting times in the queues are considered part of the times spent at the facilities, which usually becomes the costliest of the options. For example if an orbiter arrives at the OPF but has to wait for two weeks for another orbiter to finish processing in the OPF and transfer to the integration facility, those two weeks of wait are included in the total time the arriving orbiter spends at the OPF. These two weeks are also included when the total time is used in conjunction with the processing cost per day factor to generate that launches' variable cost. This logic does not accurately reflect real-world operations since the primary cost of using a facility consists of the payroll of technicians and maintenance workers that operate on the vehicle. If a vehicle is waiting for a facility, then no work is generally being performed, therefore the cost per day should not be anywhere near that of when the vehicle is currently being worked on. The RLVSim model presents users with a trigger that allows them to choose a second option that reduces the cost associated with waiting for facilities. The "Cost Incurred Outside of

Facilities" trigger on the user input form, which will be discussed in section 3.2, can be turned off ("No") so that facility waiting times are tracked separately than facility times themselves. There is also an option in the user form that allows users to select what kind of factor should be used in conjunction with the cost per day factors in order to determine what kind of cost, if any, should be incurred while vehicles wait for facilities to become available.

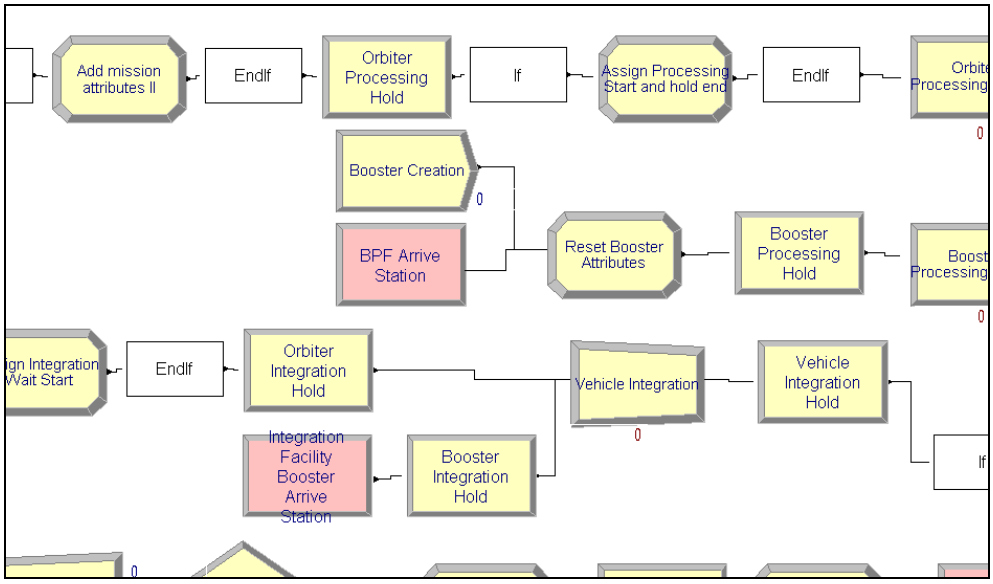


Figure 6. RLVSim Screen Capture - Integration Phase.

In order for the model to be able to track waiting times outside of the facilities and not have the waiting times included with the total facility time, separate queues were established using *Hold* modules to keep vehicles in place until the facility utilizations are less than one. A utilization of less than one allows a facility to take in another vehicle. This modeling technique renders the queues that are directly associated with the resources useless, therefore their statistical information is not contained in the simulation report. For example, orbiters waiting to use the OPF wait in the *Orbiter Processing Hold Queue* instead of the *Orbiter Processing Facility Queue*. Another reason that the logic is modeled this way is because the integration batching process that batches the orbiter and booster entities has to ensure that two incoming entities are one orbiter and one booster instead of two of one type. A *Hold* module is used to contain booster entities until an orbiter is actually in the batching module.

In order to model real-world events associated with RLVs, a loss of vehicle (LOV) decision was also included in the model that requires a user or baseline provided vehicle reliability to stochastically model catastrophic failures. This feature can be turned on and off

by a trigger in the user form and can also be further modified by identifying in the user form whether or not a replacement vehicle will be introduced in the fleet. If the replacement vehicle option is activated, a user-specified replacement time will dictate how long the fleet operates at a diminished size until a new vehicle is introduced into the model. Since RLVSim is only concerned with vehicle operation costs as opposed to development and manufacturing costs, the introduction of replacement vehicles into the system *does not* incur any extra cost. Replacement vehicles will be introduced into the system only when an LOV occurs in order to maintain the specified starting vehicle count.

In addition to the basic process flow in Figure 4, there is a section in the logic that models the off-ramp depot maintenance periods that typical RLVs go through in order to maintain vehicle health. The user has the option to turn this feature on and off in the user input form, which will dictate whether the orbiter entities enters this phase of model logic. If the trigger is turned on, then a counter is used to determine whether or not a depot maintenance visit is needed based on a user-provided missions-per-depot-maintenance field. The user also has the ability to change the cost incurred every time a depot maintenance visit is called for, as well as specify a statistical distribution (triangular) to be used to determine the duration of depot visits.

3.2 Model Input Form and General Usage Guidelines

When the model is run, a Visual Basic for Applications (VBA) coded user-form prompt asks the user which run method is desired. Figure 7 contains the first of two pages that the user has to populate in order to run the model.

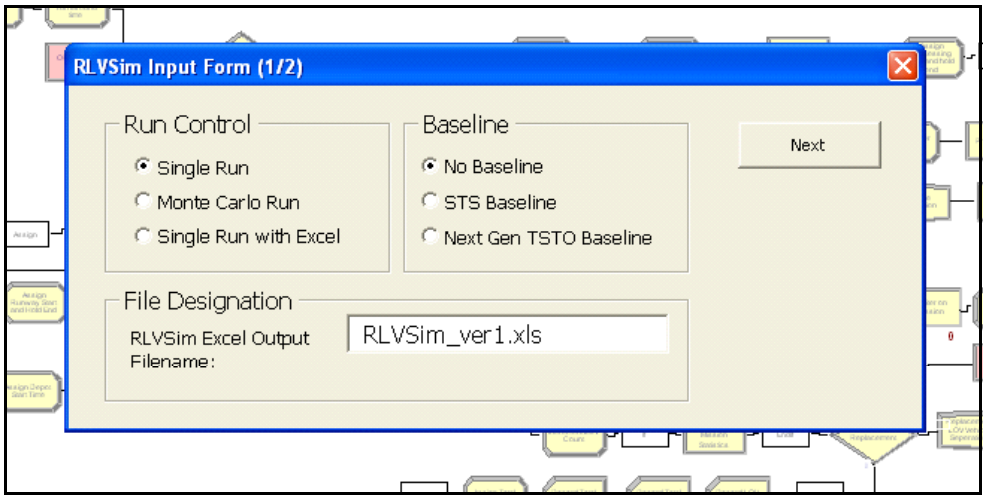


Figure 7. RLVSim Input Form – Page 1 of 2.

The "Run Control" box contains the three run types that are available. The *Single Run* option replicates the model only once and prompts the user as to whether they want to view the output report or not. The *Monte Carlo Run* option replicates the model twenty times in order to generate enough data points so that half-widths can be correctly calculated. The *Single Run with Excel* option replicates the model only once, but it exports detailed data on the first fifty launches to the Excel file specified in the *RLVSim Excel Output Filename* box. The Excel file has to be a version of the one developed for use with this model or else an error will occur.

The "Baseline" box contains three options pertaining to three different scenarios that can be used. The first scenario is a blank template where the user has to input all of the data from scratch (second page contains the vehicle input form). The second option is the current Space Transportation System (STS) baseline. The values for the inputs were generated using the STS-baseline setup of NASA's AATe. The third option pertains to an unmanned TBCC TSTO concept vehicle called *Aztec*[11] that was developed by Georgia Tech's Space Systems Design Lab (SSDL).

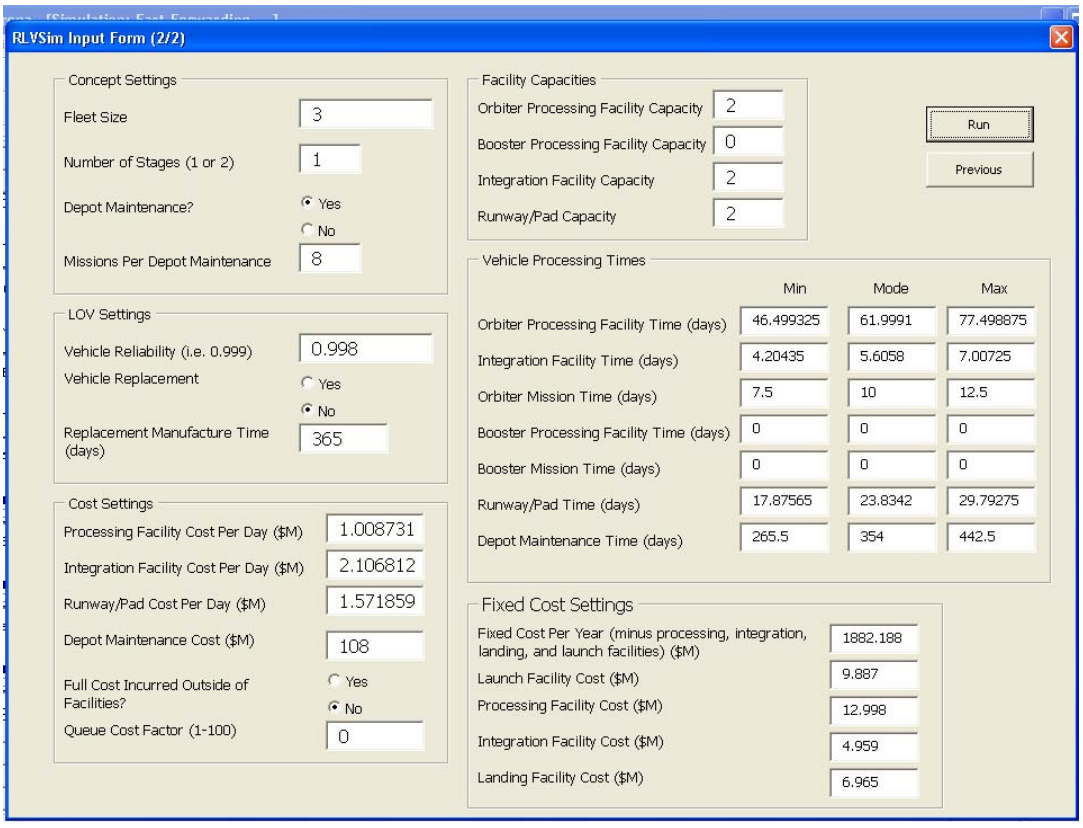


Figure 8. RLVSim Input Form – Page 2 of 2.

Figure 8 contains a screen capture of the second page of the user input form. In this particular instance the fields are populated since the STS baseline option was chosen on the prior screen. A short description of each of the fields can be seen in Table 3.

Table 3. User Input Form Fields.

Field Name	Units	Description
Fleet Size		Vehicle Fleet Size
Number of Stages		Number of vehicle stages (1 or 2)
Depot Maintenance?		Boolean trigger for off-ramp vehicle maintenance
Missions Per Depot Maintenance		Number of missions between off-ramp vehicle maintenance (if applicable)
Vehicle Reliability		Vehicle reliability in 0.xxx format
Vehicle Replacement		Boolean trigger for LOV vehicle replacement
Replacement Manufacture Time	Days	Manufacture time for LOV vehicle replacement
Processing Facility Cost Per Day	FY04 $M	Cost per day for vehicle in processing facility
Integration Facility Cost Per Day	FY04 $M	Cost per day for vehicle in integration facility
Runway/Pad Cost Per Day	FY04 $M	Cost per day for vehicle on runway/pad
Depot Maintenance Cost	FY04 $M	Off-ramp depot maintenance cost incurrence
Cost Incurred Outside of Facilities?		Boolean trigger for full-cost incurrence while waiting for facility use
Queue Cost Factor		Cost factor to cost per day rate while vehicle is waiting in queue
Orbiter Processing Facility Capacity		OPF capacity
Booster Processing Facility Capacity		BPF capacity
Integration Facility Capacity		Integration facility capacity
Runway/Pad Capacity		Runway/pad capacity
Orbiter Processing Facility Time	Days	OPF time in triangular distribution format
Integration Facility Time	Days	Integration time in triangular format
Orbiter Mission Time	Days	Orbiter mission time in triangular format
Booster Processing Facility Time	Days	BPF time in triangular format
Booster Mission Time	Days	Booster mission time in triangular format
Runway/Pad Time	Days	Runway/pad time in triangular format
Depot Maintenance Time	Days	Depot maintenance time in triangular format
Fixed Cost Minus Facility Costs	FY04 $M	Fixed per year operational costs for items such as cargo processing, traffic control, logistics, etc.
Launch Facility Cost	FY04 $M	Fixed per year cost of launch facilities
Processing Facility Cost	FY04 $M	Fixed per year cost of processing facilities
Integration Facility Cost	FY04 $M	Fixed per year cost of integration facilities
Landing Facility Cost	FY04 $M	Fixed per year cost of landing facilities

As mentioned before, there are two baselines that can be selected to use as a starting point. Both the STS and the *Aztec* are reusable launch vehicles, but they differ in the number of stages and how they are treated operationally. STS is a legacy-type system that has been used for over twenty years and was built using late 1970s technology that forces the turnaround to be approximately 90 days. *Aztec* on the other hand is a next-generation type vehicle that generally takes approximately nine days to turnaround and relaunch.

Table 4. STS and *Aztec* Baseline Settings.

Field Name	STS Baseline	*Aztec* Baseline
Fleet Size	3	2
Number of Stages (Orbiter/Booster Stages)	1	2
Depot Maintenance	Yes	Yes
Missions Per Depot Maintenance	8	8
Vehicle Reliability	0.998	0.9995
Vehicle Replacement	No	No
Replacement Manufacture Time (days)	365	100
Processing Facility Cost Per Day (FY04 $M)	1.571859	0.77815
Integration Facility Cost Per Day (FY04 $M)	1.008731	0
Runway/Pad Cost Per Day (FY04 $M)	2.106812	0.54264
Depot Maintenance Cost (FY04 $M)	108	11
Cost Incurred Outside of Facilities	No	No
Queue Cost Factor	0	0
Orbiter Processing Facility Capacity	2	20
Booster Processing Facility Capacity	0	20
Integration Facility Capacity	2	20
Runway/Pad Capacity	2	2
Orbiter Processing Facility Time (days)	TRIA(46.50, 62.00, 77.50)	TRIA(5.07, 6.76, 8.45)
Integration Facility Time (days)	TRIA(4.20, 5.61, 7.01)	0
Orbiter Mission Time (days)	TRIA(7.5, 10, 12.5)	TRIA(1.5 ,2, 2.5)
Booster Processing Facility Time (days)	0	TRIA(5.07, 6.76, 8.45)
Booster Mission Time (days)	0	TRIA(0.375, 0.5, 0.625)
Runway/Pad Time (days)	TRIA(17.88, 23.83, 29.80)	TRIA(1.71 ,2.27, 2.84)
Depot Maintenance Time (days)	TRIA(265.5 ,354, 442.5)	TRIA(26.23, 34.98, 43.72)
Fixed Cost Minus Facility Costs (FY04 $M)	1882.188	75.051
Launch Facility Cost (FY04 $M)	9.887	0.599
Processing Facility Cost (FY04 $M)	12.998	1.93
Integration Facility Cost (FY04 $M)	4.959	0
Landing Facility Cost (FY04 $M)	6.965	0.455

Table 4 contains the baseline settings for both baseline vehicles that are hard-coded into the VBA code that generates the user form. A user can choose either of these two options and change any fields needed, or can start from scratch using the "No Baseline" option. However, there are some general guidelines that have to be abided by when using the forms and general version of the model:

1. Facility capacities have to be greater than zero or else entities will queue permanently in the model (premature simulation termination). If a particular configuration does not require an integration facility or dedicated processing facility, then the distribution times for said facilities should be set to zero and the capacity set to a high value, such as 20, that will prevent the seizing of that particular resource from

becoming a bottleneck. The exception to this rule is single stage vehicles that negate the need for a booster processing facility altogether (capacity can be a zero).

2. All fields in the user forms must be populated or else a VBA error will occur that will terminate accurate data generation.
3. The Excel access mechanism will not perform properly if a copy of Excel is running in the background or foreground during the "Single Run with Excel" option. To ensure that the model runs correctly and that data is successfully populated in the Excel spreadsheet, it is recommended that the model be closed and reopened after several runs. Almost all problems regarding the model's failure to access the RLVSim spreadsheet can be corrected by closing all related programs (Arena and Excel), terminating any instances of Excel running in the background (*Processes* tab under Windows Task Manager – Ctrl-Alt-Delete), and reopening the model. The "Single Run with Excel" option actually opens the specified Excel spreadsheet before execution, so there is no need to have the RLVSim spreadsheet already open when the model is executed.
4. The "RLVSim Excel Output Filename" field has to be correctly populated with the filename of the RLVSim spreadsheet that should be located in the same folder as the model. The original filename for this spreadsheet is "RLVSim_ver1.xls", but can be changed if needed necessitating that this field be changed to the new name when the model is run in Excel output mode.
5. Modification of the Arena modules can lead to unpredictable results since many modules are specially labeled using "tags" in order to be accessed by the VBA code.

3.3 Single Run Mode

The single run mode is set up to replicate the model only once and prompt the user as to whether or not to display the report. A single run rarely generates enough data-points for Arena to be able to compute the 95% confidence intervals, so the data presented in the report is highly variable. This mode is useful if a user wishes to slow down the animation factor in order to view the entities proceeding through the model logic, which can be assisted by checking the "Highlight Active Module" option under the "Run Control" options under the "Run" option on the toolbar. Even though this mode does not generate confidence intervals on the statistics due to lack of data points, it is useful in analyses that focus on trends and sensitivities. Some trade studies that demonstrate how this option can be useful in this manner will be discussed in a later section.

3.4 Monte Carlo Mode

The Monte Carlo mode is similar to the single run mode in every way except that the model is set-up to be replicated 20 times. By replicating the model 20 times enough data points are generated for the 95% confidence intervals to be calculated. The average values that are presented in the report are averaged over each of the occurrences within a single replication (such as each launches turnaround time) as well as over all 20 replications. When the simulation is complete, the user is prompted as to whether or not to display the report.

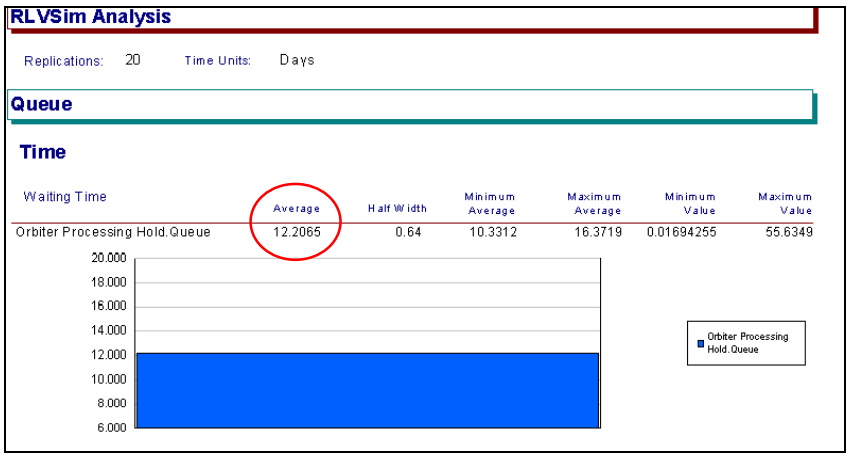


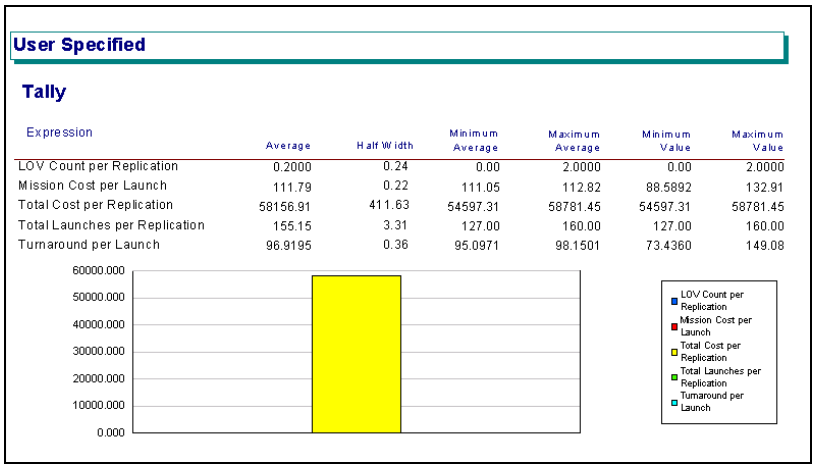
Figure 9. RLVSim Report Screenshot (Queue Statistics).

The report for this simulation is usually five pages long (using Arena version 7.01) and contains statistics regarding the Queues, the facility resources, as well as the tally averages that document the costs. Figure 9 contains a screenshot of the second page of the report, which documents some of the queue statistics. The first section documents the average waiting times associated with each of the queues. In Figure 9, only the *Orbiter Processing Hold Queue* is displayed in the top area because this is the only queue that had any entities in wait during the entire simulation. This is due to the orbiter processing time being the longest of any phase. It can be seen in the circle in Figure 9 that the orbiters wait for an average of 12.2065 days to use the OPF. This number would be higher had the fleet size been larger or smaller if the OPF had a higher capacity rate. The bottom section of the second page of the report pertains time-average number of entities waiting in the various queues.

The third, fourth, and fifth pages of the RLVSim report contain data concerning the facility resources in the model. The third page of the STS report contains an "Instantaneous Utilization" metric that is the time-weighted average of the utilization (number busy/number scheduled). There is also a "Scheduled Utilization" metric on the fourth page that is essentially the same utilization figure but uses the *average* number scheduled capacity as

opposed to the instantaneous number scheduled. Since the scheduled capacity never changes in the RLVSim model, these two metrics are the same. The third page also documents the average number busy in each of the resources. This pertains to how many units of the resources' capacity, such as the number of OPF bays, are in use on average. The fourth and fifth pages contain further documentation regarding the assigned resource capacities as well as the average number seized over the 20 replications.

The essential part of the RLVSim report is the last section, which provides tally averages for each of the five statistics that are recorded in the model. This section can be seen in Figure 10.



User Specified

Tally

Expression	Average	Half Width	Minimum Average	Maximum Average	Minimum Value	Maximum Value
LOV Count per Replication	0.2000	0.24	0.00	2.0000	0.00	2.0000
Mission Cost per Launch	111.79	0.22	111.05	112.82	88.5892	132.91
Total Cost per Replication	58156.91	411.63	54597.31	58781.45	54597.31	58781.45
Total Launches per Replication	155.15	3.31	127.00	160.00	127.00	160.00
Turnaround per Launch	96.9195	0.36	95.0971	98.1501	73.4360	149.08

Figure 10. RLVSim Report Screenshot (Tally Averages).

These five tally averages that are documented in the last section each pertain to a *Record* module placed in the model. Every time an entity passes through each of these record modules the tally is incremented by the designated amount and averaged. The three tallies that pertain to a global replication variable such as the *Total Cost per Replication* tally are recorded by creating a dummy entity at the end of the simulation (day 7300) that is immediately assigned the three global variables of interest so that the three *Record* modules the entity passes through can record the tally averages. Table 5 documents each of the five tally statistics along with a short description of each.

Table 5. RLVSim Tally Statistic Descriptions.

Expression	Units	Description
LOV Count per Replication		The average number of vehicles lost during each replication
Mission Cost per Launch	FY04 $M	The *variable* cost associated with each launch
Total Cost per Replication	FY04 $M	The total *variable* and *fixed* cost sum associated with each simulation run
Total Launches per Replication		The total number of launches per replication
Turnaround per Launch	Days	The average number of days for vehicle turnaround

The *LOV Count per Replication*, *Total Launches per Replication*, and the *Turnaround per Launch* averages are self-explanatory. The *Mission Cost per Launch* is the variable cost value associated with each launch that is calculated using the number of days in each facility along with the corresponding CPD values. This value does not take into account any fixed costs nor depot maintenance costs. The *Total Cost per Replication* encapsulates every cost associated with each replication. This is the total cost, both fixed and variable, and depot maintenance costs, of the entire 20-year simulation span. This value is updated with the variable costs after every launch, and is updated with the total fixed cost (twenty times the fixed cost per year value) at the end of each replication. A useful figure of merit is the operations cost per flight, which can be calculated by dividing the *Total Cost per Replication* by the *Total Launches per Replication* figure. This new figure will be representative of all of the costs incurred for each flight. Averaging the *Total Launches per Replication* by the 20-year simulation length also provides the average number of flights per year, which is also a commonly tracked figure of merit in LCC studies. The average number of flights per year represents the maximum achievable throughput for a vehicle for a particular fleet size and facility resource capacity. Since the Arena report generator is only capable of tallying specified statistics, the operations cost per flight and the average number of flights per year have to be calculated by hand when using RLVSim in this mode.

3.5 Single Run with Excel

The third run option that the user is presented with when running the RLVSim model is a single run with an Excel output. An Excel output file ("RLVSim_ver1.xls") was developed for Arena model to access and write detailed results pertaining to turnaround and cost metrics for the first 50 flights of the simulation run. Since a significant number of output variables are being exported from Arena and written to Excel in real-time with the model's execution, CPU memory limitations become a factor dictating that only the first 50 records be written. Any depot maintenance visits during

the first 50 flights are also recorded on a separate worksheet. Table 6 lists the twenty-two variables that are output each time an orbiter completes its turnaround operations.

Table 6. RLVSim Excel Output - Simulation Results Worksheet.

Mission Number	Processing End	Mission Start
Orbiter Number	Total Integration Time	Mission End
Mission Variable Cost	Integration Start	OPF Wait Time
Total Turnaround Time	Integration End	Integration Wait Time
Turnaround Start	Total Runway Time	Runway Wait Time
Turnaround End	Runway Start	Total Wait Time
Total Processing Time	Runway End	
Processing Start	Total Mission Time	

The *Queue Cost Trigger* in the user input form dictates how waiting times are treated. If the trigger is set to a one, then all wait time columns on the Excel output sheet will be zeroes since the waiting time will be batched with the time spent at each of the resources. If the trigger is set to a zero, then the waiting times will be populated and the mission variable cost value will be incremented appropriately based on the *Queue Cost Factor* setting. If a vehicle is lost during a mission and the *Vehicle Replacement Trigger* is set to one, then a new vehicle will appear in the schedule after the user-specified length of time. The additional vehicle will be assigned a vehicle number consistent with the number that have already been introduced into the system. For example, if vehicle two of a four-vehicle fleet is lost then the new vehicle that is introduced will be vehicle five. The following is a list of the six depot maintenance-related output variables that are written to the second tab ("Depot Maintenance Schedule") of the RLVSim Excel sheet:

- Orbiter Number
- Depot Cost
- Depot Time
- Depot Start
- Depot End
- Mission Number

The mission number that is associated with each visit is the mission immediately following the maintenance visit. The depot cost value that is populated in the sheet will always be a constant value since no distribution has been placed on depot maintenance. The Excel output sheet also has a third worksheet tab ("Average Facility Time Breakdown") that contains a pie chart depicting the time percentages of each of the major turnaround facilities

as well as the time spent in waiting. These percentages are the averages of the fifty flights that were populated in the Excel sheet.

3.6 Baseline RLVSim Results

Both baselines were evaluated using both the Monte Carlo and single run with Excel analysis options. The Excel outputs for both configurations can be seen in Appendix B. The Monte Carlo simulation results for the STS baseline are shown in Table 7.

Table 7. STS Baseline Results.

Tally Statistic	Average Value	Units
LOV Count per Replication	0.2000	
Mission Cost per Launch	111.79	FY04 $M/launch
Total Cost per Replication	58,156.91	FY04 $M
Total Launches per Replication	155.15	
Turnaround per Launch	96.9195	Days
Total Ops Cost per Mission	374.843	FY04 $M
Flights per Year	7.7575	Flights/year

The RLVSim model predicted that the total operations cost for the twenty-year simulation would be $58,157 million. When averaged over the 155.15 predicted flights this value came out to be approximately $375 million, which encapsulates all variable costs associated with the mission, the full fixed costs associated with the mission, as well as the any depot maintenance costs that are averaged into each flight. This value is consistent, but on the low end of typically quoted values for STS missions (generally $300-600 million). This can be explained by the slightly high flight rate of 7-8 flights a year. RLVSim predicts what can be achieved based on the provided fleet size but payload market demands usually dictate lower launch rate demands. Since the fixed costs are averaged over all of the flights, the total operations cost per flight figure decreases as flight rates increases. If RLVSim had predicted that STS could launch only four-five times at most based on the current fleet, then the ops cost per flight would be slightly higher and into the commonly quoted $400-500 million range.

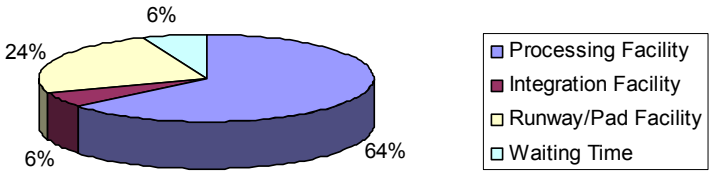


Figure 11. STS Average Facility Dwell Time Percentage Pie Chart.

Error! Reference source not found. depicts a pie chart that is representative of the percentages of each of the phases versus the overall turnaround time. It can be seen that the orbiters spend about 64% of their turnaround time in the OPFs, while only 6% of the time is spent in the integration facility. This pie chart also demonstrates that the orbiters spend about 6% of their time waiting for one of the facilities to become available.

The *Aztec* baseline resulted in a total ops cost of $8,724.84 million for the entire 20-year simulation (Table 8). This cost translates to a $9.6370 million cost when divided over the 905 flights. *Aztec* certainly achieves a much higher flight rate than STS because it takes on average 9.6 days to be readied for relaunch. Because of the high flight rate, the average LOV count for *Aztec* is higher than that of the STS baseline despite *Aztec* having a higher reliability. An *Aztec* fleet size of two can achieve a maximum of approximately 45 flights per year based on the baseline assumptions.

Table 8. *Aztec* Baseline Results.

Tally Statistic	Average Value	Units
LOV Count per Replication	0.4500	
Mission Cost per Launch	6.4963	FY04 $M/launch
Total Cost per Replication	8,724.84	FY04 $M
Total Launches per Replication	905.35	
Turnaround per Launch	9.2654	Days
Total Ops Cost per Mission	9.6370	FY04 $M
Flights per Year	45.268	Flights/year

It can be seen in Figure 12 that the *Aztec* orbiters spend about 71% of their time in the processing facility while spending only 24% of their time on the runway/pad. Only 5% of the time is spent in waiting due to baseline assumptions that *Aztec* has a three-vehicle fleet with a dual-processing facility capability. The primary different between the *Aztec* facility percentage pie chart and that of STS (**Error! Reference source not found.**) is that *Aztec* spends no time in any sort of integration facility, which is consistent with the model's assumption that *Aztec* is integrated on the runway/pad.

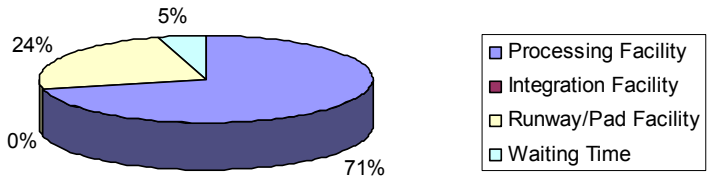


Figure 12. *Aztec* Orbiter Average Facility Dwell Time Percentage Pie Chart.

4.0 RLVSim ModelCenter® Capability

ModelCenter® is a multidisciplinary design suite that allows users to incorporate many different types of tools (Excel, Fortran code, etc.) into one framework in order to parametrically study complex designs. Using VBScript code in a ModelCenter® ScriptWrapper, Arena models can be integrated into the framework in order to be used along with other tools. For instance, an Arena model's variable values, expression values, process distribution time, etc. can all be updated with results from an Excel-based tool in a rapid manner that allows various trade studies or even numerical optimizations to be performed. Such a VBScript ScriptWrapper was written for the RLVSim model in order to be able to perform trade studies, such as optimizing fleet size based on facility capacities. The ScriptWrapper utilizes the ActiveX® Automation[6] technology to allow ModelCenter® to manipulate not just the various modules inside the RLVSim model but also the Arena run controls that dictate the running of the simulation. When Arena is installed on a computer, its object library is added to the list of object libraries registered with the local Windows system. VBScript codes should then be able to utilize the Visual Basic commands associated with the object library, and typical Windows applications like Excel can do so through VBA. In fact, the user input forms that are presented to the user when the simulation model is executed were built using Excel VBA code that utilizes the Arena object library.

4.1 RLVSim ScriptWrapper Details

The code for the ModelCenter® ScriptWrapper ("RLVSim_ver1.ScriptWrapper") can be fully seen in Appendix C. It is written in similar fashion to the user input form in that the user has a choice between three options (a new concept, *Aztec*, or STS*)*. The primary difference is that if a user selects *Aztec* or STS, the only ModelCenter® input variables that can be changed and still influence the model are the fleet size and facility capacity values. The other model input-specific values become obsolete when a "ConceptChoice" of 2 (STS) or 3 (*Aztec*) is chosen. The script is setup this way to allow users to perform trade studies on either of the two baselines regarding fleet size versus facility capability settings. The first choice, the new concept choice, is set up by default to have STS values. The user is free to change any of the inputs to see the impact on the resulting output metrics. The first choice is also used in conjunction with the RLVSim version for AATe that has been modified from the original AATe to output RLVSim specific parameters such as the cost per day and fixed facility cost values. The ScriptWrapper outputs the same tally statistics as the RLVSim report, as well as an *AverageCostPerYear* and *OpsCostPerFlight* metric. The *AverageCostPerYear* value is the average of the total cost over the 20-year span. The *OpsCostPerFlight* is the total

cost incurred, fixed and variable, for each flight that is calculated by dividing the total cost by the number of flights achieved.

The ScriptWrapper specifically controls the RLVSim model by opening the RLVSim model file, updating all of the fields that would normally be updated by the user input form, and fast-forwarding the simulation. Once the simulation is complete the tally statistics are retrieved from their SIMAN-related model records and the model is saved and closed. It should be noted that several changes need to be made to the RLVSim model before the ScriptWrapper will successfully run the model:

- While open, the model's report generator has to be set to "never" so that the report prompt will not interfere with ModelCenter® execution. This can be achieved by specifying this particular option on the "Reports" tab in the "Run Setup" options (under "Run" on toolbar). This change does not have to be reset in order for normal operation of the RLVSim model since the VBA user input form sets it appropriately.
- The user input form has to be turned off in order for the ModelCenter® ScriptWrapper to operate correctly. This is achieved by clicking on the "Basic Process" module template, choosing the "Variable" module, clicking on the "Initial Value" box on the "User Form Trigger" variable, and setting the value to zero. This change will have to be reversed before normal operation of the RLVSim model since it prevents the user input form from being prompted when the simulation is run.
- Likewise, the "Excel Output Trigger" has to be set to a zero to prevent the model from writing to the RLVSim Excel file during ModelCenter® use. This change will be reset any time the model is run with the "User Form Trigger" set to a one.

4.2 RLVSim & AATe Integration

The modified version of AATe (AATe_RLVSim_ver.xls) and the AATe ModelCenter® ExcelWrapper were modified to allow ModelCenter® to receive the RLVSim specific values from the output page. The output variables of interest are tabulated in Table 9.

Table 9. RLVSim Related AATe Output Variables.

Output Name	Units	Description
LaunchfacilityTime_cf	Days	Curve-fit launch facility time
LandingFacilityTime_cf	Days	Curve-fit landing facility time
ProcessingFacilityTime_cf	Days	Curve-fit processing facility time
IntegrationFacilityTime_cf	Days	Curve-fit integration facility time
BaseFixedOpsCost	FY04 $M	Fixed cost per year minus fixed facility maintenance cost
FixedLaunchFacilityCost	FY04 $M	Fixed launch facility cost per year
FixedLandingFacilityCost	FY04 $M	Fixed landing facility cost per year
FixedTurnaroundFacilityCost	FY04 $M	Fixed turnaround facility cost per year
FixedIntegrationFacilityCost	FY04 $M	Fixed integration facility cost per year
LaunchCPD	FY04 $M/Day	Launch facility cost per day per year
TurnaroundCPD	FY04 $M/Day	Turnaround facility cost per day
IntegrationCPD	FY04 $M/Day	Integration facility cost per day

A user wishing utilize AATe in conjunction with the RLVSim model in ModelCenter® will have to establish links between the variables in Table 9 with the proper inputs in the RLVSim ScriptWrapper as well as any other variable that they may wish to have fed from AATe to RLVSim, such as a fleet size.

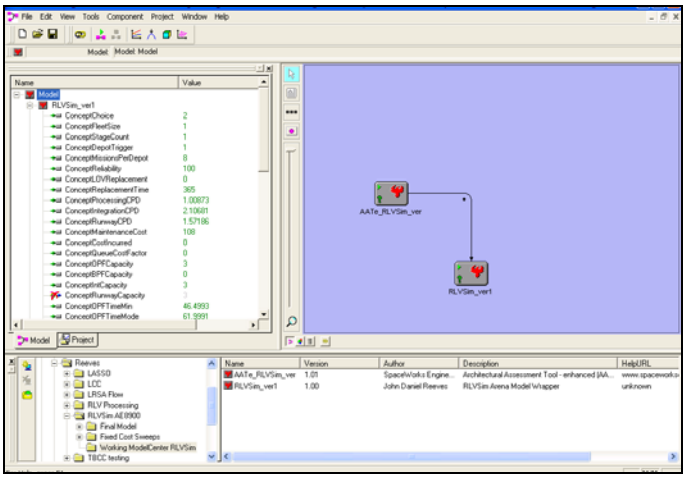


Figure 13. AATe and RLVSim in ModelCenter®.

4.3 *Aztec* ModelCenter® Trade Study

A typical RLV design study would consist of a conceptual level LCC analysis to determine the economic viability of the vehicle. Based on what missions a particular RLV is designed for, an analysis of the ability of the architecture to achieve various mission goals for a reasonable cost is required. One major driver in such a study is the market demand, whether it is commercial or governmental, of payloads that the RLV is capable of delivering. From this demand a desired annual launch rate can be estimated for a RLV concept. AATe provides the general turnaround values for various phases of an RLV's preparation for flight, but this does not take into account facility resource constraints and facility queues. How many orbiter processing facilities are needed? How many launch pads would allow the desired launch rate to be achieved? These questions can be answered by using RLVSim in the ModelCenter® environment. It should be noted that each simulation run in ModelCenter® requires roughly ten seconds to return resulting values, so smaller scale analyses such as these do not present a problem in terms of run times.

Aztec has a turnaround time of around nine days according to AATe, but how does that translate to an increased fleet size along with a specified facility resource level? If the payload delivery market will support 100 launches per year, how large should the *Aztec* fleet size be? Trade studies performed using RLVSim shed insight to these questions by demonstrating the general trends associated with increasing fleet sizes based on various facility resource designations.

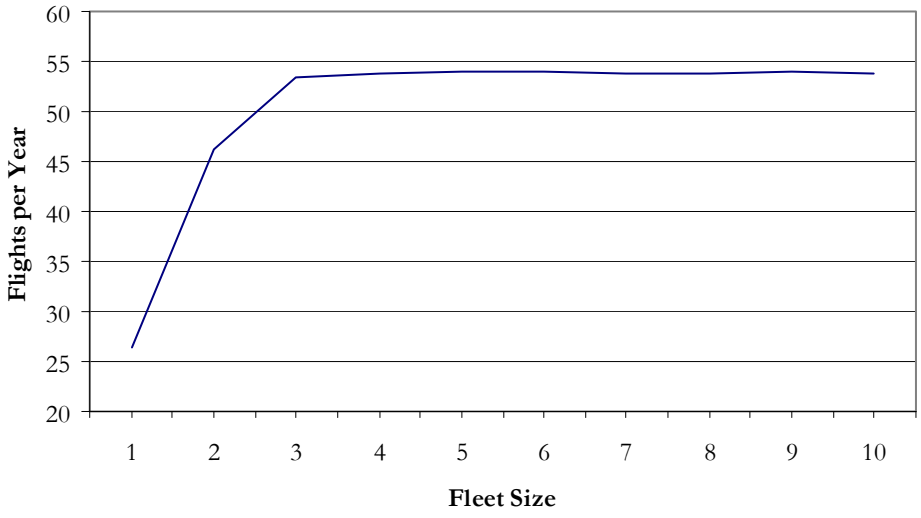


Figure 14. *Aztec* Flight Rate vs. Fleet Size with Facility Capacity at 1.

The first phase of the study was performed by sweeping *Aztec's* fleet size from one to ten while holding each of the facility resource levels (OPF, BPF, and runway/pad) at one. Because *Aztec* is processed in dedicated processing facilities and integrated on the runway/pad, only the OPF, BPF, and runway pad resource facilities were taken into account. It is apparent from Figure 14 that a flight rate of 100 flights per year could never be achieved if the facility capacity levels were only one, regardless of the number of vehicles in the fleet. In fact, adding more than three vehicles would not be beneficial since bottlenecks in the processing flow would prevent any increase in flight rate beyond 53-54 flights per year.

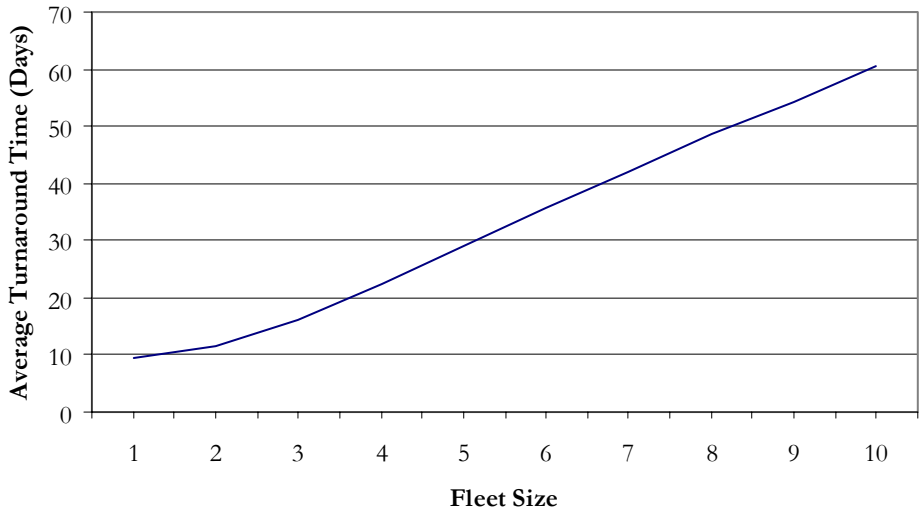


Figure 15. *Aztec* Turnaround Time vs. Fleet Size with Facility Capacity at 1.

Figure 15 demonstrates that the average turnaround time for the vehicles increases almost linearly as more vehicles are added to the fleet. This is expected since additional vehicles would spend a majority of their time waiting to enter a facility. If the market demands 100 flights per year, then it is apparent that a higher facility resource level is needed.

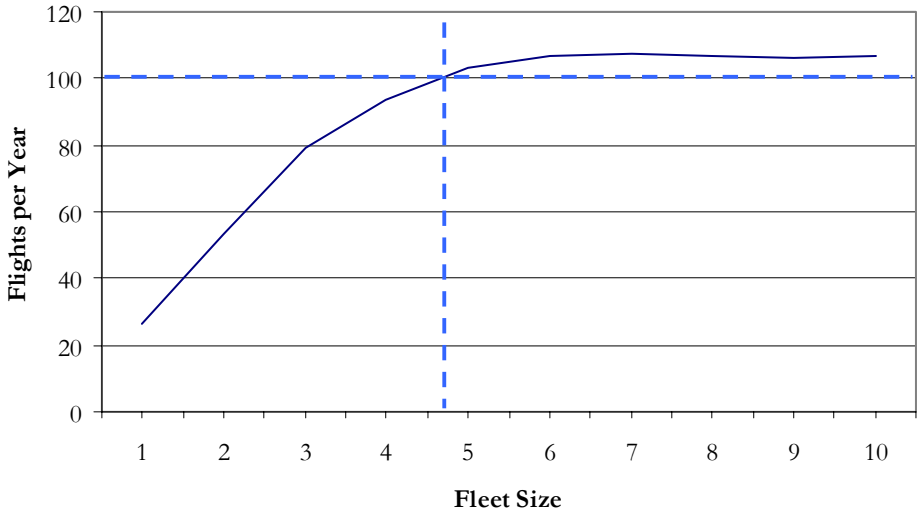


Figure 16. *Aztec* Flight Rate vs. Fleet Size with Facility Capacity at 2.

Figure 16 was created by sweeping the fleet size from one to ten while holding all of the facility resource levels at two. A level of two can be considered a "dual-processing" capability that significantly increases the maximum throughput of the process. Figure 16 indicates that a flight rate of 100 flights per year *is* achievable if there are at least five vehicles in the fleet. The desired launch rate is obtained by this fleet size, but only by a small margin. It is apparent that a dual-processing capability still allows only a maximum of around 110 flights per year, which is not significantly higher than the target of 100 per year. Because of this, to achieve confidence that the vehicle architecture is capable of always achieving the desired flight rate, a sixth vehicle may or may not be required.

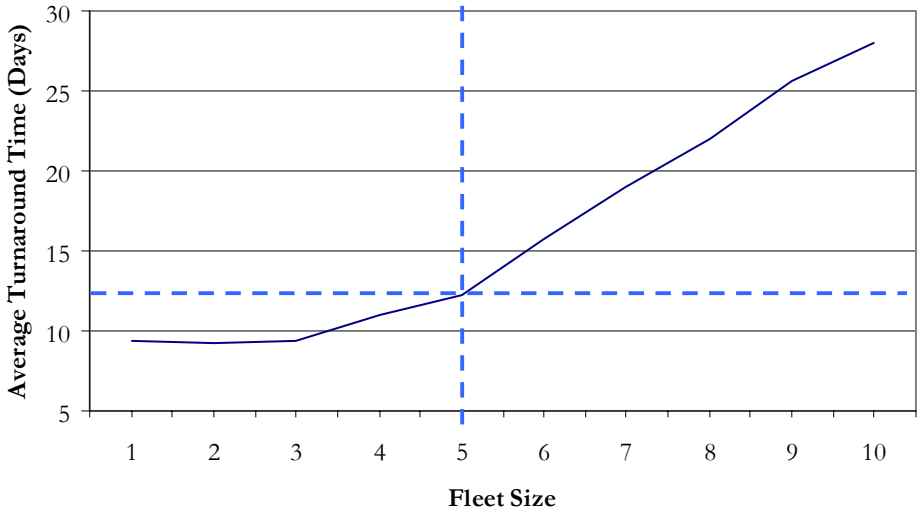


Figure 17. *Aztec* Turnaround Time vs. Fleet Size with Facility Capacity at 2.

If the fleet size is determined to be five vehicles, an average turnaround time of 12-13 days should be expected according to Figure 17. The average turnaround curve increases almost linearly if more than five vehicles are introduced which is consistent with the leveling-out of the achievable flights per year plot in Figure 16.

A dual-processing capability along with five vehicles seems to be adequate, but in order for the architecture to be robust enough to handle fluctuating yearly launch demands, it may be wise to increase the facility resource capability one step further.

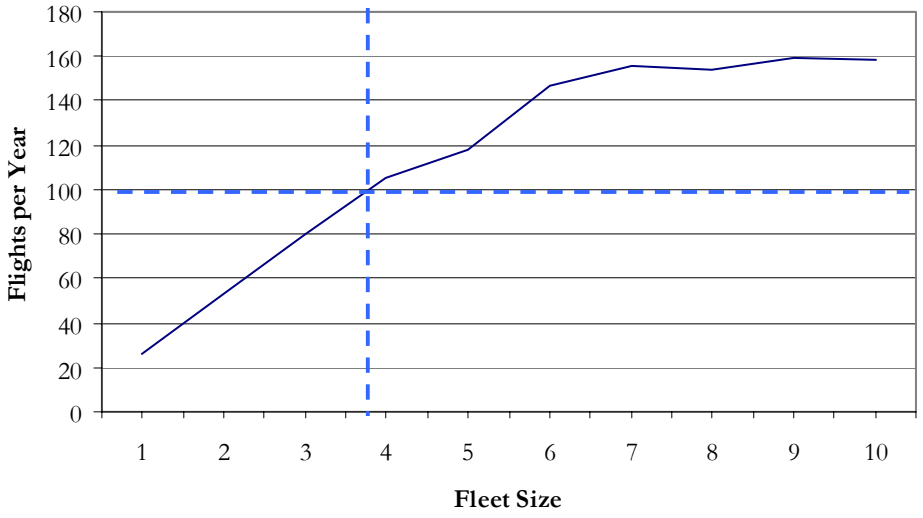


Figure 18. *Aztec* Flight Rate vs. Fleet Size with Facility Capacity at 3.

Figure 18 demonstrates that a facility resource capacity level of three may be a better choice because the trend levels-out at a much higher flight rate than the dual-capacity scenario. The five-vehicle fleet should obtain a flight rate of 120 flights per year, while a four-vehicle fleet would still achieve the 100 flights per year target. If only four vehicles were used, the average turnaround per vehicle would be just over nine days according to Figure 19.

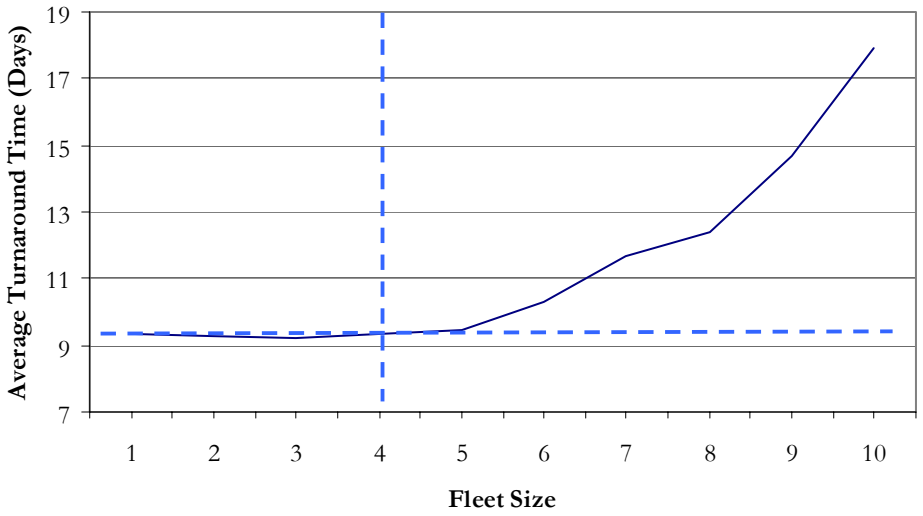


Figure 19. *Aztec* Turnaround Time vs. Fleet Size with Facility Capacity at 3

It appears that the target flight rate could be achieved in either of two ways. First, a dual-processing facility capacity could be used in conjunction with a five-vehicle fleet. An alternative would be to use only a four-vehicle fleet but with a facility resource capacity level of three. Table 8 contains the results from each of the scenarios.

Table 10. *Aztec* Trade Study Results.

Processing Capability Level	Fleet Size to Meet 100 flights/year	Approximate Achievable Launch Rate	Average Turnaround (Days) at Optimal Fleet Size
1	N/A	54	N/A
2	5	108	12.5
3	4	155	9.5

The two alternative solutions to the problem should be compared in terms of financial commitment. Generally, the cheaper alternative would be to use the dual-processing capability with the five vehicles since adding an extra vehicle will usually be cheaper than adding more facility infrastructure as long as the vehicle is not too complex. Not only would the expanded infrastructure require a significant cost commitment up front for construction, but the yearly maintenance costs would be higher as well as the costs associated with the labor force needed to operate the facilities. However, Figure 16 shows that having five vehicles with a dual-processing capability barely gets an achievable throughput that is higher than the target. This is important to keep in mind, because any detriment to the fleet, such as a LOV scenario or longer than normal vehicle refurbishment, may lead to a flight rate lower than needed. If the infrastructure is kept at a facility capacity level of two, it may be wise in include a sixth vehicle. Even with an additional vehicle, the achievable throughput may not be sufficient if the demand fluctuates. In order to be truly prepared for a market fluctuation scenario more facilities may be needed. Despite the additional costs, a facility resource capacity of three would allow a more robust response to any fluctuations, and would allow a fleet size of four or five ensure that the targeted flight rate of 100 flights per year be achieved.

4.4 STS ModelCenter® Trade Study

The current STS fleet consists of three vehicles, and the processing and launch facilities at the Kennedy Space Center can generally process two vehicles in parallel. Using these baseline assumptions in RLVSim, a maximum launch rate of almost eight flights a year can be achieved. STS initially was designed to have a much higher flight rate, and modifications and additions to the International Space Station (ISS) necessitate that STS achieve as high a flight rate as possible. Because of this, it may be of interest to see what

changes could be made to improve the possible throughput. Basically, how much of a benefit can be achieved if each of the facilities were improved to handle three vehicles at a time? In addition, what kind of benefit would a fourth replacement orbiter provide if the infrastructure were expanded?

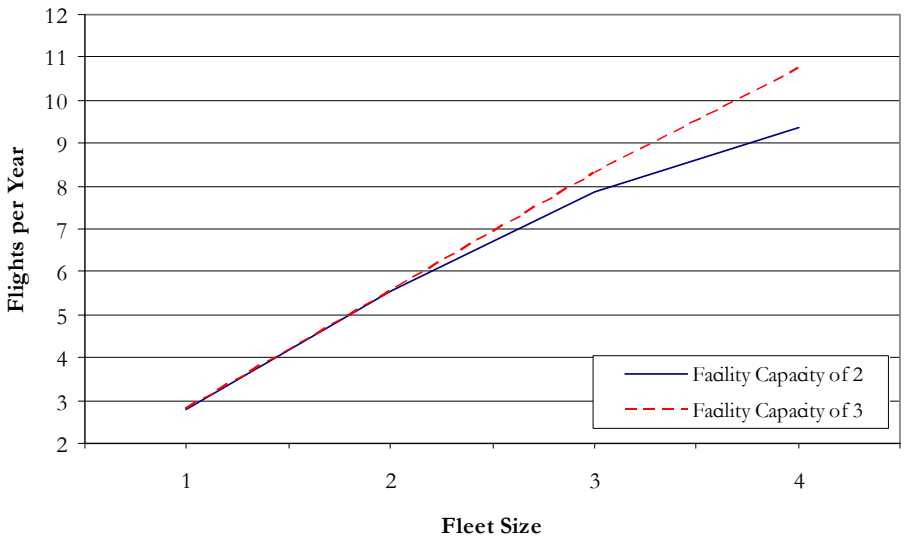


Figure 20. STS Flights per Year vs. Fleet Size.

Figure 20 was generated using the STS baseline assumptions in RLVSim along with ModelCenter® in a similar manner as the *Aztec* trade study in section 4.3. ModelCenter was used to sweep the fleet size from one to four to see the effects on the flight rate and turnaround time based on specified facility resource levels. The results show that a slight improvement in the flight rate can be achieved if the infrastructure is expanded. With three vehicles this improvement is very slight, going from just under 8 to around 8.5 flights a year. If a fourth vehicle was added to replace the orbiter Columbia, the modified infrastructure would have a more significant benefit. Four vehicles operating in the infrastructure as it is currently configured would achieve just over 9 flights a year. Four vehicles operating in an expanded infrastructure would achieve almost 11 flights per year. Figure 21 shows that the average turnaround time decreases from the two-vehicle processing capability to the three-vehicle processing capability. In this case a four-orbit fleet would see a 118-day turnaround time reduced to 95 days if the infrastructure were expanded.

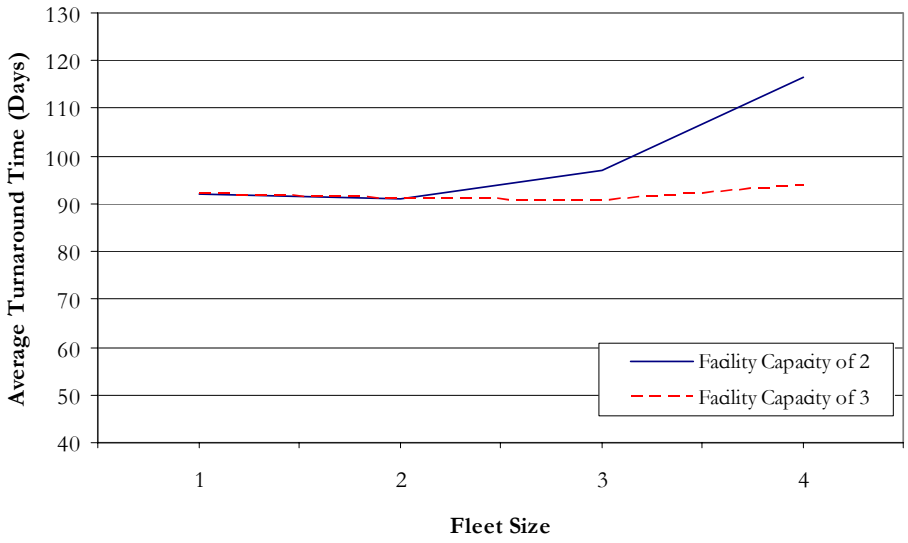


Figure 21. STS Turnaround Time vs. Fleet Size.

The STS is a legacy vehicle that is significantly close to retirement. Throughout the years many proposals have been put forth regarding ways to improve the fleet in order to keep it flying for many more years, but the recent change of focus in the space industry to Lunar and Martian exploration has restricted future plans of STS operation to within the current decade. Because of this, the STS trade study documented in this section is solely provided to give an example of how RLVSim can be used in conjunction with ModelCenter® to study the impacts of improvements to existing architectures that are in place today. Even though the STS fleet may not be used after 2010, a next generation vehicle may take advantage of the STS processing and launch infrastructure at KSC, just as STS took advantage of what was left from the Apollo program of the 1960s.

5.0 Conclusions

In summery RLVSim is a tool that can be used in several different ways to assist in LCC analyses of single-stage and two-stage reusable launch vehicles. In its primary role as a stochastic simulator that can be replicated many times to obtain confidence intervals associated with each of the output metrics, RLVSim provides a capability to students and student researchers that is significantly different from traditional deterministic tools. Statistical support for any forecasted value is beneficial since it captures the variability associated with such a value. In its secondary role as a tool that can collaborate with other tools in the ModelCenter® environment, RLVSim can quickly provide results based on various input settings. These settings allow quick trade studies to be performed using RLVSim, and allows the tool to act as a contributing analysis to larger scale concept studies.

Section 3.6 documented both the STS and *Aztec* baseline results which demonstrate the validity of RLVSim's outputs. Sections 4.3 and 4.4 each described typical analyses that can be performed using RLVSim, demonstrating the importance of decisions that can be made using RLVSim regarding turnaround drivers such as infrastructure processing levels and fleet sizes. There are various input parameters to RLVSim, any of which could be identified as trade study subjects to truly discern what factors lead to high operational costs.

Because RLVSim was developed to be used in several different fashions, there are a number of supporting files other than the model itself that need to be accounted for. Table 11 provides a brief description all of the files that have been developed for this project, all of which are needed to operate RLVSim along with AATe in the ModelCenter® environment.

Table 11. RLVSim Related Files and Spreadsheets.

File Name	Description
RLVSim_ver1.doe	RLVSim Arena model
RLVSim_ver1.xls	RLVSim Excel output spreadsheet
AATe_RLVSim_ver.xls	AATe Excel tool modified for RLVSim use
RLVSim_ver1.ScriptWrapper	ModelCenter® ScriptWrapper file for RLVSim_ver1.doe file
AATe_RLVSim_ver.ExcelWrapper	ModelCenter® ExcelWrapper for RLVSim_ver1.xls file

In the effort to truly bring operational considerations to the forefront of conceptual vehicle design, RLVSim assists by complementing other tools commonly used such as RMAT and AATe. It has been developed in order to provide sophisticated results while keeping in mind that the average operations researcher may not be familiar with the intricacies of DES. The user forms allow even beginner users to tweak time and cost

parameters for each of the major turnaround facilities used in a vehicle's turnaround flow, which leads to a high level of control over the concept of operations formulation. Due to this versatility, RLVSim is a prime example of how discrete event simulation can be brought into a complex vehicle's conceptual design, allowing probabilistic and dynamic analyses to be performed. These analyses offer insight into vehicle turnaround operations that have not been possible with tools in the past.

References

1. CAIB, "Columbia Accident Investigation Board: Report I," August 2003.
2. Isakowitz, S.J., Hopkins, J.P., Hopkins, J.B., *International Reference Guide to Space Launch Systems,* Third Edition, AIAA, 1999.
3. Unal, R., Morris, W.D., White, N., Lepsch, R., Brown, R., "Approximation model building for reliability and maintainability characteristics of reusable launch vehicles," AIAA-2000-4712, 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach, California, September 6-8, 2000.
4. KSC Next Gen Site, (science.ksc.nasa.gov/shuttle/nexgen/AATe_main.htm), Accessed July 26, 2004.
5. GEMFLO, KSC Next Gen Site, (science.ksc.nasa.gov/shuttle/nexgen/GEM-FLO_main.htm), Accessed July 26, 2004.
6. Kelton, W. D., Sadowski, R., Sturrock, D., *Simulation with Arena*, Third Edition, McGraw-Hill, New York, New York, 2004.
7. Rabadi, G., private communication, Old Dominion University, Norfolk, Virginia, June 6, 2004.
8. Arena Simulation webpage, (www.arenasimulation.com), Accessed July 21, 2004.
9. Banks, J., Carson III, J., Nelson, B., *Discrete-Event System Simulation*, Second Edition, Prentice Hall, Upper Saddle River, New Jersey, 1996.
10. Hayter, A., *Probability and Statistics for Engineers and Scientists*, 2nd Edition, Duxbury, Pacific Grove, California, 2002.
11. Kokan, T., Olds, J., Hutchinson, V., Reeves, J.D., "Aztec: A TSTO Hypersonic Vehicle Concept Utilizing TBCC and HEDM Propulsion Technologies," AIAA-2004-3728, 40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, Fort Lauderdale, Florida, July 11-14, 2004.
12. Cost Estimating Web Site, NASA, (www.jsc.nasa.gov/bu2/), Accessed July 15, 2004.
13. Morris, W.D., private communication, "OMDP Strategy_LessonsLeaned.doc", NASA Langley, Hampton, Virginia, July 14, 2004.

Appendix A – Baseline Data Derivations

Both the STS and *Aztec* baseline configurations that are in the model are primarily based on results from NASA's AATe. A modified version of AATe that automatically calculates the variable cost per day per each phase of the turnaround operations as well as the fixed cost values was constructed. Some of the same assumptions used in the AATe models were utilized in the RLVSim model, such as orbiter time in orbit values, turnaround flow paths, and nominal fleet sizes.

A.1 Space Transportation System Baseline

The baseline STS configuration assumes a single-stage orbiter component since the space shuttle itself is the primary piece of the configuration. In reality, there are two solid rocket boosters (SRBs) and one external tank (ET) that are integrated into the stack, but since these components are either expendable or partially reusable, they do not drive the turnaround time operations. While detailed studies of the current Space Transportation System derived systems would require detailed DES-type analysis of SRB and ET arrival, inspection, and transfer, the RLVSim model has been developed to be used for next generation RLV conceptual studies that will not require expendable boosters or tanks. The RLVSim model also had to be kept under educational-mode constraints, which prevented worthwhile modeling of the SRB and ET type components.

AATe's output spreadsheet provides a variable cost breakdown that specifies the values associated with the three phases of interest: processing, integration, and pad/runway. These are three out of four phases that AATe uses to roll-up overall turnaround time, with the fourth being the landing phase. Since the landing phase does not contribute significantly to the overall turnaround time (less that 1%), its AATe generated time and variable cost contributions were combined into those of the runway/pad phase. In order to quantify a cost per day factor for each of the modeled phases, the variable cost contribution of each phase was first translated to a percentage of the sum of the three phases of interest (processing + integration + launch/landing). For example, AATe calculated that the variable cost per launch of STS due to the processing phase was $6.37 million, which is around 56% of the variable cost sum of the three major phases ($3.82 M + $6.37 M + $1.20 M = $11.39 M). The percentages of each of the three phases were then used to spread the remaining variable cost per flight - traffic control costs, cargo processing costs, logistics costs, etc. - amongst the launch/landing, processing, and integration phases. For the processing phase this resulted in a total variable processing cost of $56.80 M. These total variable costs were then used in conjunction with the curve-fit turnaround times in AATe (still assuming launch and landing are combined) in order to generate a variable cost per day

factor (CPD) for each of the phases. These CPD figures were then translated from FY00 dollars to FY04 dollars using an inflation factor (1.101) that was taken from the "NASA New Start Index Inflation Calculator"[12]. Figure contains a screen-shot directly from the modified version of AATe that contains the CPD calculations for the STS baseline.

RLVSim Output Metrics

	cost ($M)	%	time (days)	add-on ($M)	total cost ($M)	per day	FY04$M
launch	3.82	0.335054	**23.8342**	30.21073573	34.03	1.427664	**1.571859**
turnaround	6.37	0.559322	**61.99914**	50.43223513	56.80	0.916195	**1.008731**
integration	1.20	0.105624	**5.605769**	9.523752551	10.73	1.913544	**2.106812**
total	11.39		91.4391				
remaining	90.167						

Launch	23.01129
Landing	0.82291
turnaround	61.99914
integration	5.605769
	91.4391

Figure A-1. STS Cost per Day Calculations (from modified AATe).

The fixed cost values used to populate the STS baseline were also pulled directly from AATe, this time by separating the fixed "launch", "landing", "turnaround", and "integration" operations cost values from the total fixed operations cost per year in the "Detailed Operations Outputs" section. The remaining $1882.188 million ($1,709.526 FY00 M) was then used as the baseline fixed cost per year value, which is populated in the user input form. AATe predicts that this baseline fixed cost per year value starts to increase when a launch rate greater than 12 per year is achieved. Such a rate is out of the range of a three vehicle STS fleet (the RLVSim model predicts that the fleet can achieve a maximum 8 flights per year). When the model is run, the facility costs that were removed from the total fixed operations cost per year are added back into the *TotalFixedCostPerYear* model variable based on the number of facilities designated in the user form. For example, the baseline STS configuration is established with two processing bays, two integration bays, and two launch facilities (along with an assumed single runway). The fixed cost per year value that is populated into the model during execution is calculated by adding: $1882.188 M base value + $6.965 M (landing facility) + 2 x $9.887 M (launch facility) + 2 x $12.998 M (processing facility) + 2 x $4.959 M (integration facility) = $1,944.841 M (all dollar figures FY04). At the end of each replication 20 times this cost (20 years) is added into the total cost of the twenty-year simulation run in order to capture the cost associated with facility overhead, logistics support, and so forth.

The curve-fit turnaround times that were used to populate the model were pulled directly from the AATe curve-fit values. The launch and landing curve-fit time values were combined into the one value for the model, and a +/- factor of 25% was used to obtain the minimum, mode, and maximum values of triangular distributions for each of the three phases. A factor of 25% was chosen since it allows enough variation to impact the throughput, but is still consistent with the nominal value. The ranges were chosen to be symmetric since the nominal value itself is an estimation, and it is not known whether the value is skewed towards the minimum or maximum.

The depot maintenance value used for the STS baseline was derived from a reference[13] that listed the depot maintenance times for three shuttle flights. These times should be representative of the average range of time spent in depot maintenance. The three quoted depot maintenance visits (STS-89, 101, 109) are from a six-year time span and show an increasing trend in both the number of days and the associated cost of the visit. However, these three depot maintenance visits took place at the Palmdale facility in California, which has since discontinued such work. The orbiter depot maintenance now takes place in the third OPF at the Kennedy Space Center, thus requiring a less time and cost to complete. Since the value used in the RLVSim model is just a basis of a statistical distribution, it is believed that the *average* (with inflation taken into account on the cost value) of the three historical depot maintenance visit values is adequate. An assumption being made is that the increasing trend in depot maintenance time will be counteracted by having the maintenance operations transferred back to the Kennedy Space Center. Table A-1 contains the turnaround time in days and the associated cost for the depot maintenance for each of the three flights.

Table A-1. Depot Maintenance Reference Values[14].

Mission Number	Depot Maintenance Time (days)	Depot Maintenance Cost ($M)	Depot Maintenance Cost (FY04 $M)
STS-89	235	49 (FY98)	57
STS-101	310	88 (FY00)	97
STS-109	517	162 (FY02)	169
average	**354**		**108**

The RLVSim model was populated with a nominal STS depot maintenance time of 354 days, which is the average calculated in Table . In order to determine the nominal depot maintenance cost, each of the three cited dollar figures were translated to FY2004 dollars and averaged. This computation resulted in a value of $108 M. Just as in the case with the processing, integration, and pad/runway phases, a +/- 25% margin was added to the nominal turnaround value in order to obtain a triangular distribution where the minimum is

265.5 days and the maximum is 442.5 days. The nominal value of 354 days itself was used as the mode in the distribution.

A.2 *Aztec* Baseline

The *Aztec* baseline was computed in similar fashion to that of the STS baseline using NASA's AATe. The primary difference is that *Aztec*'s turnaround flow consists of only the launch and processing phases since all vehicle integration takes place on the pad. Figure A-2 contains a screen-shot from the modified version of AATe that documents how the variable cost per day factors were calculated.

RLVSim Output Metrics

	cost	%	time	add-on	total cost	per day	FY04
launch	0.22	0.189966	**2.273686**	0.898741	1.12	0.492861	**0.54264**
turnaround	0.95	0.810034	**6.760915**	3.83233	4.78	0.70677	**0.778153**
integration	0.00	0	**0**	0	0.00	0	**0**
total	1.17		9.034601				
remaining	4.731						

Launch	1.611962
Landing	0.052765
turnaround	6.760915
integration	0.60896
	9.034601

Figure A-2. *Aztec* Cost per Day Calculations (from modified AATe).

As shown in Figure A-2, the integration variable cost per flight figure was zero for *Aztec*. Because *Aztec* is integrated on the pad, AATe assumes that the integration costs are encapsulated in the variable launch cost per flight. AATe does break the turnaround time down into the four major phases (launch, landing, processing, and integration), but the modified version of AATe combines the launch, landing, and integration values into the launch turnaround time in order to calculate the variable launch cost per day factor. This factor was calculated as being $0.492861 M/day, but became $0.54264 M/day when the 1.101 inflation factor was applied. The variable processing cost per day factor was calculated as being $0.70677 M/day, or $0.778153 M/day in FY04 dollars.

The fixed costs per year values for the *Aztec* baseline were calculated in the exact same way as the STS baseline, though the integration related costs were zero since *Aztec* is integrated on the runway. A fixed cost minus facility maintenance costs of $75.051 M ($68.166 FY00 M) was derived with the assumption that *Aztec* will never have a fleet size more than two vehicles. A fleet of this size translates to a maximum flight rate of 45.3 per year in the RLVSim model. The fixed cost minus facility cost value in AATe does not start to increase beyond a constant value of $75.051 M until a flight rate greater than 60 flights per

year is achieved. At the beginning of an RLVSim *Aztec* run, the pad/runway capacity entered by the user is used to add the fixed facility maintenance costs back into the *TotalFixedCostPerYear* value that is populated into the model.

The nominal depot maintenance time and cost for *Aztec* were calculated using the ratio of the total AATe turnaround of *Aztec* to that of STS along with the nominal STS depot maintenance values derived in section A.1. A nominal maintenance time of 34.9765 days and a nominal maintenance cost of $11 M were derived from the following relationships:

$$\left(\frac{9.0346\_days}{91.44\_days}\right) * 354\_days = 34.9765\_days \tag{A-1}$$

and

$$\left(\frac{9.0346\_days}{91.44\_days}\right) * \$108\_million = \$11\_million \tag{A-2}$$

The 34.9765 day nominal depot maintenance time was then used to develop a triangular distribution to be used in the model by using a +/- 25% range to determine the min and the max value. The nominal value itself was used as the mode in this distribution.

Appendix B – Baseline Excel Outputs

Table B-1. STS RLVSim Excel Output Schedule.

RLVSim Output Spreadsheet																					
Mission Number	Orbiter Number	Mission Variable Cost (FY04 $M)	Total Turnaround Time (days)	Turnaround Start (day)	Turnaround End (day)	Total Processing Time (days)	Processing Start (day)	Processing End (day)	Total Integration Time (days)	Integration Start (day)	Integration End (day)	Total Runway Time (days)	Runway Start (day)	Runway End (day)	Total Mission Time (Days)	Mission Start (day)	Mission End (day)	OPF Wait Time (days)	Integration Wait Time (days)	Runway Wait Time (days)	Total Wait Time (days)
1	1	108.36	88.64	0.00	88.64	59.64	0.00	59.64	4.88	59.64	64.52	24.12	64.52	88.64	0.46	88.64	89.10	0.00	0.00	0.00	0.00
2	2	106.97	84.48	13.85	98.33	51.80	13.85	65.64	6.25	65.64	71.89	26.44	71.89	98.33	0.41	98.33	98.74	0.00	0.00	0.00	0.00
3	3	118.64	134.42	24.66	159.07	72.24	59.64	131.88	5.66	131.88	137.54	21.53	137.54	159.07	0.50	159.07	159.57	34.98	0.00	0.00	34.98
4	1	132.91	108.51	89.10	197.61	72.85	89.10	161.95	6.30	161.95	168.25	29.36	168.25	197.61	0.39	197.61	198.00	0.00	0.00	0.00	0.00
5	2	112.18	123.33	98.74	222.07	58.44	131.88	190.32	6.22	190.32	196.54	25.53	196.54	222.07	0.38	222.07	222.45	33.14	0.00	0.00	33.14
6	3	123.42	103.98	159.57	263.55	70.17	161.95	232.12	6.07	232.12	238.19	25.36	238.19	263.55	0.46	263.55	264.01	2.39	0.00	0.00	2.39
7	1	107.14	88.16	198.00	286.16	61.38	198.00	259.38	5.85	259.38	265.23	20.93	265.23	286.16	0.45	286.16	286.62	0.00	0.00	0.00	0.00
8	2	109.30	98.91	222.45	321.36	60.58	232.12	292.70	5.87	292.70	298.57	22.80	298.57	321.36	0.40	321.36	321.76	9.67	0.00	0.00	9.67
9	3	116.64	95.88	264.01	359.90	66.77	264.01	330.79	6.58	330.79	337.37	22.53	337.37	359.90	0.39	359.90	360.29	0.00	0.00	0.00	0.00
10	1	116.35	100.10	286.62	386.71	61.19	292.70	353.89	5.68	353.89	359.57	27.14	359.57	386.71	0.43	386.71	387.14	6.08	0.00	0.00	6.08
11	2	117.97	107.14	321.76	428.90	69.78	330.79	400.56	5.68	400.56	406.24	22.66	406.24	428.90	0.47	428.90	429.38	9.03	0.00	0.00	9.03
12	3	117.42	96.31	360.29	456.59	65.85	360.29	426.14	5.83	426.14	431.97	24.62	431.97	456.59	0.33	456.59	456.92	0.00	0.00	0.00	0.00
13	1	110.99	104.52	387.14	491.66	62.22	400.56	462.78	5.29	462.78	468.07	23.59	468.07	491.66	0.43	491.66	492.09	13.42	0.00	0.00	13.42
14	2	119.43	99.93	429.38	529.31	71.98	429.38	501.36	5.39	501.36	506.75	22.56	506.75	529.31	0.40	529.31	529.71	0.00	0.00	0.00	0.00
15	3	97.58	86.94	456.92	543.86	57.74	462.78	520.52	4.97	520.52	525.48	18.37	525.48	543.86	0.42	543.86	544.28	5.86	0.00	0.00	5.86
16	1	103.25	93.51	492.09	585.60	57.25	501.36	558.60	5.74	558.60	564.35	21.25	564.35	585.60	0.49	585.60	586.09	9.27	0.00	0.00	9.27
17	2	101.43	81.19	529.71	610.90	52.12	529.71	581.83	5.91	581.83	587.74	23.16	587.74	610.90	0.38	610.90	611.28	0.00	0.00	0.00	0.00
18	3	120.65	115.29	544.28	659.57	72.99	558.60	631.59	5.70	631.59	637.29	22.28	637.29	659.57	0.44	659.57	660.01	14.32	0.00	0.00	14.32
19	1	128.17	106.72	586.09	692.80	75.28	586.09	661.37	5.26	661.37	666.63	26.18	666.63	692.80	0.44	692.80	693.24	0.00	0.00	0.00	0.00
20	2	112.38	115.52	611.28	726.80	70.39	631.59	701.98	4.40	701.98	706.38	20.42	706.38	726.80	0.37	726.80	727.18	20.31	0.00	0.00	20.31
21	3	126.44	107.31	660.01	767.33	76.70	661.37	738.06	5.76	738.06	743.82	23.51	743.82	767.33	0.45	767.33	767.77	1.35	0.00	0.00	1.35
22	1	125.57	113.57	693.24	806.81	75.83	701.98	777.81	6.52	777.81	784.32	22.49	784.32	806.81	0.40	806.81	807.21	8.74	0.00	0.00	8.74
23	2	106.94	97.89	727.18	825.07	58.10	738.06	796.17	5.41	796.17	801.57	23.50	801.57	825.07	0.41	825.07	825.48	10.88	0.00	0.00	10.88
24	3	98.89	87.73	767.77	855.50	47.72	777.81	825.52	6.81	825.52	832.33	23.17	832.33	855.50	0.43	855.50	855.93	10.03	0.00	0.00	10.03
25	2	124.49	102.82	1135.11	1237.92	71.06	1135.11	1206.17	5.41	1206.17	1211.58	26.35	1211.58	1237.92	0.41	1237.92	1238.34	0.00	0.00	0.00	0.00
26	1	116.65	94.02	1189.12	1283.15	60.69	1189.12	1249.82	5.68	1249.82	1255.49	27.66	1255.49	1283.15	0.33	1283.15	1283.48	0.00	0.00	0.00	0.00
27	2	102.59	83.41	1238.34	1321.75	56.33	1238.34	1294.67	5.99	1294.67	1300.66	21.09	1300.66	1321.75	0.44	1321.75	1322.19	0.00	0.00	0.00	0.00
28	3	114.84	91.07	1251.45	1342.52	56.65	1251.45	1308.10	6.71	1308.10	1314.81	27.71	1314.81	1342.52	0.45	1342.52	1342.98	0.00	0.00	0.00	0.00
29	1	125.17	114.53	1283.48	1398.01	71.45	1294.67	1366.12	5.56	1366.12	1371.68	26.33	1371.68	1398.01	0.45	1398.01	1398.45	11.19	0.00	0.00	11.19
30	2	104.14	86.41	1322.19	1408.60	60.89	1322.19	1383.09	4.89	1383.09	1387.97	20.62	1387.97	1408.60	0.41	1408.60	1409.00	0.00	0.00	0.00	0.00
31	3	109.74	114.18	1342.98	1457.15	64.05	1366.12	1430.17	5.08	1430.17	1435.25	21.91	1435.25	1457.15	0.41	1457.15	1457.56	23.14	0.00	0.00	23.14
32	1	102.30	83.61	1398.45	1482.06	56.31	1398.45	1454.76	4.84	1454.76	1459.60	22.46	1459.60	1482.06	0.47	1482.06	1482.53	0.00	0.00	0.00	0.00
33	2	107.48	108.93	1409.00	1517.94	59.65	1430.17	1489.82	5.82	1489.82	1495.64	22.29	1495.64	1517.94	0.38	1517.94	1518.32	21.17	0.00	0.00	21.17
34	3	110.97	88.51	1457.56	1546.08	55.84	1457.56	1513.41	6.14	1513.41	1519.55	26.53	1519.55	1546.08	0.44	1546.08	1546.52	0.00	0.00	0.00	0.00
35	1	113.62	99.44	1482.53	1581.97	62.03	1489.82	1551.85	6.91	1551.85	1558.76	23.21	1558.76	1581.97	0.42	1581.97	1582.39	7.29	0.00	0.00	7.29
36	2	116.92	96.95	1518.32	1615.27	68.14	1518.32	1586.45	5.41	1586.45	1591.87	23.40	1591.87	1615.27	0.51	1615.27	1615.77	0.00	0.00	0.00	0.00
37	3	115.80	99.24	1546.52	1645.77	61.72	1551.85	1613.57	5.49	1613.57	1619.05	26.71	1619.05	1645.77	0.37	1645.77	1646.14	5.33	0.00	0.00	5.33
38	1	101.53	86.06	1582.39	1668.46	53.24	1586.45	1639.69	4.89	1639.69	1644.58	23.88	1644.58	1668.46	0.42	1668.46	1668.88	4.06	0.00	0.00	4.06
39	2	110.11	91.10	1615.77	1706.88	63.71	1615.77	1679.48	5.20	1679.48	1684.68	22.19	1684.68	1706.88	0.37	1706.88	1707.25	0.00	0.00	0.00	0.00
40	3	107.80	89.84	1646.14	1735.97	63.65	1646.14	1709.79	4.56	1709.79	1714.35	21.62	1714.35	1735.97	0.39	1735.97	1736.36	0.00	0.00	0.00	0.00
41	1	110.52	100.09	1668.88	1768.97	59.00	1679.48	1738.48	5.76	1738.48	1744.24	24.73	1744.24	1768.97	0.40	1768.97	1769.36	10.60	0.00	0.00	10.60
42	2	126.33	105.36	1707.25	1812.61	68.56	1709.79	1778.35	6.22	1778.35	1784.57	28.04	1784.57	1812.61	0.44	1812.61	1813.05	2.54	0.00	0.00	2.54
43	3	113.57	94.38	1736.36	1830.75	61.08	1738.48	1799.56	5.49	1799.56	1805.05	25.70	1805.05	1830.75	0.43	1830.75	1831.18	2.11	0.00	0.00	2.11
44	1	99.85	89.72	1769.36	1859.08	53.48	1778.35	1831.83	5.75	1831.83	1837.58	21.50	1837.58	1859.08	0.43	1859.08	1859.51	8.99	0.00	0.00	8.99
45	2	102.55	82.97	1813.05	1896.01	54.86	1813.05	1867.91	5.66	1867.91	1873.56	22.45	1873.56	1896.01	0.34	1896.01	1896.35	0.00	0.00	0.00	0.00
46	3	110.66	89.81	1831.18	1920.99	58.12	1831.83	1889.96	6.08	1889.96	1896.03	24.95	1896.03	1920.99	0.36	1920.99	1921.35	0.66	0.00	0.00	0.66
47	1	94.47	85.34	1859.51	1944.84	52.39	1867.91	1920.30	5.69	1920.30	1925.99	18.85	1925.99	1944.84	0.40	1944.84	1945.24	8.40	0.00	0.00	8.40
48	3	93.70	75.35	1921.35	1996.70	48.55	1921.35	1969.90	4.86	1969.90	1974.76	21.94	1974.76	1996.70	0.42	1996.70	1997.12	0.00	0.00	0.00	0.00
49	2	130.35	106.93	2264.57	2371.49	72.33	2264.57	2336.89	5.62	2336.89	2342.51	28.98	2342.51	2371.49	0.34	2371.49	2371.83	0.00	0.00	0.00	0.00
50	1	113.79	92.30	2290.39	2382.69	61.31	2290.39	2351.69	6.03	2351.69	2357.72	24.97	2357.72	2382.69	0.39	2382.69	2383.08	0.00	0.00	0.00	0.00

Table B-2. STS RLVSim Excel Depot Maintenance Schedule.

Orbiter Number	Depot Cost per Visit (FY04 $M)	Depot Time (days)	Depot Start (day)	Depot End (day)	Mission Number
2.00	108.00	309.63	825.48	1135.11	23.00
1.00	108.00	381.92	807.21	1189.12	22.00
3.00	108.00	395.52	855.93	1251.45	24.00
2.00	108.00	368.21	1896.35	2264.57	45.00
1.00	108.00	345.14	1945.24	2290.39	47.00
3.00	108.00	394.17	1997.12	2391.29	48.00

Table B-3. *Aztec* RLVSim Excel Output Schedule.

RLVSim Output Spreadsheet																						
Mission Number	Orbiter Number	Variable Cost (FY04 $M)	Total Turnaround Time (days)	Turnaround Start (day)	Turnaround End (day)	Processing Time (days)	Processing Start (day)	Processing End (day)	Total Integration Time (days)	Integration Start (day)	Integration End (day)	Runway Time (days)	Runway Start (day)	Runway End (day)	Mission Time (Days)	Mission Start (day)	Mission End (day)	Wait Time (days)	Integration Wait Time (days)	Wait Time (days)	Wait Time (days)
1	1	5.47	8.48	0.00	8.48	5.65	0.00	5.65	0.00	6.50	6.50	1.98	6.50	8.48	0.08	8.48	8.57	0.00	0.86	0.00	0.86
2	1	5.78	9.58	8.57	18.15	5.94	8.57	14.51	0.00	16.01	16.01	2.14	16.01	18.15	0.08	18.15	18.23	0.00	1.50	0.00	1.50
3	2	7.55	12.26	10.81	23.07	7.94	10.81	18.75	0.00	20.55	20.55	2.52	20.55	23.07	0.09	23.07	23.16	0.00	1.80	0.00	1.80
4	1	7.29	10.11	18.23	28.34	7.65	18.23	25.88	0.00	25.88	25.88	2.46	25.88	28.34	0.09	28.34	28.43	0.00	0.00	0.00	0.00
5	2	5.28	8.62	23.16	31.78	5.39	23.16	28.55	0.00	29.79	29.79	2.00	29.79	31.78	0.09	31.78	31.87	0.00	1.23	0.00	1.23
6	1	7.04	9.70	28.43	38.12	7.54	28.43	35.96	0.00	35.96	35.96	2.16	35.96	38.12	0.09	38.12	38.22	0.00	0.00	0.00	0.00
7	2	6.19	8.73	31.87	40.61	6.47	31.87	38.34	0.00	38.48	38.48	2.13	38.48	40.61	0.10	40.61	40.71	0.00	0.14	0.00	0.14
8	1	7.27	10.06	38.22	48.28	7.70	38.22	45.92	0.00	45.92	45.92	2.36	45.92	48.28	0.09	48.28	48.37	0.00	0.00	0.00	0.00
9	2	6.68	9.37	40.71	50.08	6.78	40.71	47.49	0.00	47.49	47.49	2.59	47.49	50.08	0.08	50.08	50.16	0.00	0.00	0.00	0.00
10	1	6.13	8.53	48.37	56.90	6.38	48.37	54.74	0.00	54.74	54.74	2.15	54.74	56.90	0.06	56.90	56.96	0.00	0.00	0.00	0.00
11	2	6.31	9.18	50.16	59.35	6.50	50.16	56.66	0.00	57.04	57.04	2.31	57.04	59.35	0.10	59.35	59.44	0.00	0.37	0.00	0.37
12	1	5.92	9.25	56.96	66.21	5.68	56.96	62.64	0.00	63.45	63.45	2.76	63.45	66.21	0.10	66.21	66.31	0.00	0.81	0.00	0.81
13	2	6.03	9.23	59.44	68.67	6.22	59.44	65.67	0.00	66.49	66.49	2.18	66.49	68.67	0.08	68.67	68.75	0.00	0.82	0.00	0.82
14	1	6.94	9.46	66.31	75.77	7.68	66.31	73.98	0.00	73.98	73.98	1.79	73.98	75.77	0.07	75.77	75.84	0.00	0.00	0.00	0.00
15	2	6.63	9.62	68.75	78.36	6.95	68.75	75.70	0.00	76.12	76.12	2.24	76.12	78.36	0.09	78.36	78.45	0.00	0.42	0.00	0.42
16	2	6.18	8.62	78.45	87.07	6.38	78.45	84.83	0.00	84.83	84.83	2.24	84.83	87.07	0.08	87.07	87.15	0.00	0.00	0.00	0.00
17	1	5.80	8.12	111.75	119.88	5.91	111.75	117.67	0.00	117.67	117.67	2.21	117.67	119.88	0.09	119.88	119.96	0.00	0.00	0.00	0.00
18	1	6.59	9.27	119.96	129.24	6.62	119.96	126.58	0.00	126.58	126.58	2.65	126.58	129.24	0.08	129.24	129.31	0.00	0.00	0.00	0.00
19	2	6.20	8.59	121.41	130.00	6.52	121.41	127.94	0.00	127.94	127.94	2.07	127.94	130.00	0.10	130.00	130.10	0.00	0.00	0.00	0.00
20	1	5.47	9.70	129.31	139.01	5.34	129.31	134.65	0.00	136.59	136.59	2.42	136.59	139.01	0.08	139.01	139.09	0.00	1.94	0.00	1.94
21	2	7.27	10.14	130.10	140.24	7.86	130.10	137.96	0.00	138.12	138.12	2.12	138.12	140.24	0.09	140.24	140.33	0.00	0.16	0.00	0.16
22	1	5.70	9.37	139.09	148.46	5.89	139.09	144.99	0.00	146.40	146.40	2.06	146.40	148.46	0.10	148.46	148.56	0.00	1.42	0.00	1.42
23	2	5.88	8.27	140.33	148.59	6.10	140.33	146.43	0.00	146.50	146.50	2.09	146.50	148.59	0.09	148.59	148.69	0.00	0.07	0.00	0.07
24	1	5.98	8.26	148.56	156.82	6.37	148.56	154.93	0.00	154.93	154.93	1.89	154.93	156.82	0.10	156.82	156.92	0.00	0.00	0.00	0.00
25	2	6.28	9.19	148.69	157.88	6.32	148.69	155.01	0.00	155.38	155.38	2.50	155.38	157.88	0.09	157.88	157.97	0.00	0.37	0.00	0.37
26	1	6.44	9.73	156.92	166.65	6.73	156.92	163.65	0.00	164.43	164.43	2.23	164.43	166.65	0.09	166.65	166.74	0.00	0.78	0.00	0.78
27	2	6.02	9.97	157.97	167.94	5.81	157.97	163.77	0.00	165.18	165.18	2.76	165.18	167.94	0.09	167.94	168.03	0.00	1.41	0.00	1.41
29	2	6.03	8.39	168.03	176.42	6.28	168.03	174.30	0.00	174.30	174.30	2.12	174.30	176.42	0.09	176.42	176.51	0.00	0.00	0.00	0.00
28	1	7.21	10.09	166.74	176.84	7.84	166.74	174.59	0.00	174.79	174.79	2.04	174.79	176.84	0.08	176.84	176.92	0.00	0.21	0.00	0.21
30	2	5.99	9.23	176.51	185.74	6.33	176.51	182.84	0.00	183.78	183.78	1.96	183.78	185.74	0.08	185.74	185.82	0.00	0.94	0.00	0.94
31	1	6.22	9.88	176.92	186.80	6.44	176.92	183.36	0.00	184.57	184.57	2.23	184.57	186.80	0.09	186.80	186.88	0.00	1.21	0.00	1.21
32	2	7.00	9.69	185.82	195.51	7.39	185.82	193.21	0.00	193.21	193.21	2.29	193.21	195.51	0.08	195.51	195.58	0.00	0.00	0.00	0.00
33	1	6.87	9.60	221.72	231.32	7.08	221.72	228.80	0.00	228.80	228.80	2.52	228.80	231.32	0.10	231.32	231.42	0.00	0.00	0.00	0.00
34	2	6.44	8.81	223.99	232.81	7.03	223.99	231.02	0.00	231.02	231.02	1.78	231.02	232.81	0.08	232.81	232.89	0.00	0.00	0.00	0.00
35	1	5.80	8.79	231.42	240.21	5.87	231.42	237.29	0.00	237.93	237.93	2.28	237.93	240.21	0.09	240.21	240.30	0.00	0.64	0.00	0.64
36	2	7.24	9.92	232.89	242.81	7.89	232.89	240.77	0.00	240.77	240.77	2.03	240.77	242.81	0.09	242.81	242.90	0.00	0.00	0.00	0.00
37	1	5.58	10.13	240.30	250.43	5.46	240.30	245.76	0.00	247.99	247.99	2.44	247.99	250.43	0.08	250.43	250.51	0.00	2.23	0.00	2.23
38	2	7.08	9.77	242.90	252.66	7.57	242.90	250.47	0.00	250.47	250.47	2.20	250.47	252.66	0.09	252.66	252.75	0.00	0.00	0.00	0.00
39	1	6.38	8.93	250.51	259.44	6.50	250.51	257.01	0.00	257.01	257.01	2.43	257.01	259.44	0.10	259.44	259.54	0.00	0.00	0.00	0.00
40	2	5.41	7.84	252.75	260.60	5.37	252.75	258.12	0.00	258.32	258.32	2.28	258.32	260.60	0.10	260.60	260.69	0.00	0.20	0.00	0.20
41	1	6.42	9.08	259.54	268.62	6.87	259.54	266.40	0.00	266.63	266.63	1.99	266.63	268.62	0.08	268.62	268.70	0.00	0.23	0.00	0.23
42	2	6.41	9.06	260.69	269.75	6.83	260.69	267.52	0.00	267.73	267.73	2.03	267.73	269.75	0.08	269.75	269.84	0.00	0.21	0.00	0.21
43	1	6.87	9.43	268.70	278.12	7.44	268.70	276.13	0.00	276.13	276.13	1.99	276.13	278.12	0.09	278.12	278.22	0.00	0.00	0.00	0.00
44	2	6.32	9.91	269.84	279.75	6.55	269.84	276.39	0.00	277.49	277.49	2.26	277.49	279.75	0.08	279.75	279.83	0.00	1.10	0.00	1.10
45	1	6.93	9.96	278.22	288.17	7.12	278.22	285.34	0.00	285.61	285.61	2.56	285.61	288.17	0.08	288.17	288.26	0.00	0.27	0.00	0.27
46	2	6.98	9.55	279.83	289.38	7.64	279.83	287.47	0.00	287.47	287.47	1.91	287.47	289.38	0.08	289.38	289.46	0.00	0.00	0.00	0.00
47	1	6.71	9.76	288.26	298.01	7.08	288.26	295.34	0.00	295.80	295.80	2.21	295.80	298.01	0.09	298.01	298.10	0.00	0.47	0.00	0.47
48	2	7.48	10.32	289.46	299.79	7.99	289.46	297.45	0.00	297.45	297.45	2.33	297.45	299.79	0.08	299.79	299.87	0.00	0.00	0.00	0.00
49	2	6.33	8.90	332.19	341.09	6.38	332.19	338.57	0.00	338.57	338.57	2.52	338.57	341.09	0.09	341.09	341.17	0.00	0.00	0.00	0.00
50	1	6.61	9.18	337.46	346.64	6.92	337.46	344.38	0.00	344.38	344.38	2.26	344.38	346.64	0.08	346.64	346.72	0.00	0.00	0.00	0.00

Table B-4. *Aztec* RLVSim Excel Depot Maintenance Schedule.

Orbiter Number	Depot Cost per Visit (FY04 $M)	Depot Time (days)	Depot Start (day)	Depot End (day)	Mission Number
1.00	11.00	35.91	75.84	111.75	14.00
2.00	11.00	34.26	87.15	121.41	16.00
1.00	11.00	34.84	186.88	221.72	31.00
2.00	11.00	28.41	195.58	223.99	32.00
2.00	11.00	32.32	299.87	332.19	48.00
1.00	11.00	39.35	298.10	337.46	47.00

Appendix C – RLVSim ScriptWrapper Code

```
#
# This is a ScriptWrapper for the RLVSim Arena Discrete Event Simulation Model
#
# @author: John Daniel Reeves
# @description: RLVSim Arena Model Wrapper
# @version: 1.00
# @date: 7/10/2004

# general variables

variable: ConceptChoice                 double  input   description="1 - new concept, 2 - STS, 3 - Aztec"          lowerbound=1   upperbound=3
variable: ConceptFleetSize              double  input   description="Orbiter and Booster fleet size"              lowerbound=1
variable: ConceptStageCount             double  input   description="Number of stages (1 or 2)"                   lowerbound=1   upperbound=2
variable: ConceptDepotTrigger           double  input   description="Depot maintenance trigger. 0=No 1=Yes"      lowerbound=0   upperbound=1
variable: ConceptMissionsPerDepot       double  input   description="Number of missions between depot maintenance"
variable: ConceptReliability            double  input   description="Vehicle reliability in 0.xxx format"
variable: ConceptLOVReplacement         double  input   description="Trigger for LOV Replacement, 0=No 1=Yes"  lowerbound=0  upperbound=1
variable: ConceptReplacementTime        double  input   units="days"  description="LOV Replacement time in applicable"
variable: ConceptProcessingCPD          double  input   units="$M"   description="Cost per day for processing facility"
variable: ConceptIntegrationCPD         double  input   units="$M"   description="Cost per day for integration facility"
variable: ConceptRunwayCPD              double  input   units="$M"   description="Cost per day for Runway/Pad facility"
variable: ConceptMaintenanceCost        double  input   units="$M"   description="Average cost for depot maintenance"
variable: ConceptCostIncurred           double  input   description="Trigger for full cost penalty while in queues, 0=No 1=Yes"   lowerbound=0  upperbound=1
variable: ConceptQueueCostFactor        double  input   description="Cost factor for vehicles in queue 0-100%"      lowerbound=0
variable: ConceptOPFCapacity            double  input   description="Orbiter Processing Facility capacity"
variable: ConceptBPFCapacity            double  input   description="Booster Processing Facility capacity"
variable: ConceptIntCapacity            double  input   description="Integration facility capacity"
variable: ConceptRunwayCapacity         double  input   description="Launch runway/pad facility capacity"
variable: ConceptOPFTimeMin             double  input   units="days"  description="OPF time triangular distribution min"
variable: ConceptOPFTimeMode            double  input   units="days"  description="OPF time triangular distribution mode"
variable: ConceptOPFTimeMax             double  input   units="days"  description="OPF time triangular distribution max"
variable: ConceptBPFTimeMin             double  input   units="days"  description="BPF time triangular distribution min"
variable: ConceptBPFTimeMode            double  input   units="days"  description="BPF time triangular distribution mode"
variable: ConceptBPFTimeMax             double  input   units="days"  description="BPF time triangular distribution max"
variable: ConceptIntMin                 double  input   units="days"  description="Integration time triangular distribution min"
variable: ConceptIntMode                double  input   units="days"  description="Integration time triangular distribution mode"
variable: ConceptIntMax                 double  input   units="days"  description="Integration time triangular distribution max"
variable: ConceptOrbiterMissionMin      double  input   units="days"  description="Orbiter mission time triangular distribution min"
variable: ConceptOrbiterMissionMode     double  input   units="days"  description="Orbiter mission time triangular distribution mode"
variable: ConceptOrbiterMissionMax      double  input   units="days"  description="Orbiter mission time triangular distribution max"
variable: ConceptBoosterMissionMin      double  input   units="days"  description="Booster mission time triangular distribution min"
variable: ConceptBoosterMissionMode     double  input   units="days"  description="Booster mission time triangular distribution mode"
variable: ConceptBoosterMissionMax      double  input   units="days"  description="Booster mission time triangular distribution max"
variable: ConceptRunwayMin              double  input   units="days"  description="Launch runway/pad time triangular distribution min"
variable: ConceptRunwayMode             double  input   units="days"  description="Launch runway/pad time triangular distribution mode"
variable: ConceptRunwayMax              double  input   units="days"  description="Launch runway/pad time triangular distribution max"
variable: ConceptDepotMin               double  input   units="days"  description="Depot Maintenance time triangular distribution min"
variable: ConceptDepotMode              double  input   units="days"  description="Depot Maintenance time triangular distribution mode"
variable: ConceptDepotMax               double  input   units="days"  description="Depot Maintenance time triangular distribution max
variable: ConceptBaseFixedCost          double  input   units="$M"   description="Fixed cost per year minus facility maintenance costs"
variable: ConceptLaunchFixedCost        double  input   units="$M"   description="Launch facility related fixed costs"
variable: ConceptLandingFixedCost       double  input   units="$M"   description="Landing facility related fixed costs"
variable: ConceptProcessingFixedCost    double  input   units="$M"   description="Processing facility related fixed costs"
variable: ConceptIntegrationFixedCost   double  input   units="$M"   description="Integration facility related fixed costs"

variable: LaunchCount                   double  output              description="Number of launches during 20 year simulation"
variable: AverageTurnaround             double  output units="days" description="Average turnaround during 20 year simulation"
variable: AverageLaunchCost             double  output units="$M"   description="Average launch cost during 20 year simulation"
variable: TotalCost                     double  output units="$M"   description="Total cost of launches during 20 year simulation"
variable: LOVCount                      double  output              description="LOV count during 20 year simulation"
variable: LaunchesPerYear               double  output units="Launches/yr"  description="Average number of launches per year"
variable: AverageCostPerYear            double  output units="$M"   description="Average variable cost per year"
variable: OpsCostPerFlight              double  output units="$M/flight"    description="Total cost per flight (variable + fixed)"

#-------------------------------------------------------------------------------
script:

'create application variables
dim AM
dim excel

' set default values to STS
ConceptFleetSize = 3
ConceptStageCount = 1
ConceptDepotTrigger = 1
ConceptMissionsPerDepot = 8
ConceptReliability = 0.998
ConceptLOVReplacement= 0
ConceptReplacementTime = 365
ConceptProcessingCPD = 1.008731
ConceptIntegrationCPD = 2.106812
ConceptRunwayCPD = 1.571859
ConceptMaintenanceCost = 108
ConceptCostIncurred = 0
ConceptQueueCostFactor = 0
ConceptOPFCapacity = 2
```

```
ConceptBPFCapacity = 0
ConceptIntCapacity = 2
ConceptRunwayCapacity = 2
ConceptOPFTimeMin = 46.499325
ConceptOPFTimeMode = 61.9991
ConceptOPFTimeMax = 77.498875
ConceptIntMin = 4.20435
ConceptIntMode = 5.6058
ConceptIntMax = 7.00725
ConceptOrbiterMissionMin = 7.5
ConceptOrbiterMissionMode = 10
ConceptOrbiterMissionMax = 12.5
ConceptBPFMin = 0
ConceptBPFMode = 0
ConceptBPFMax = 0
ConceptBoosterMissionTimeMin = 0
ConceptBoosterMissionTimeMode = 0
ConceptBoosterMissionTimeMax = 0
ConceptRunwayMin = 17.87565
ConceptRunwayMode = 23.8342
ConceptRunwayMax = 29.79275
ConceptDepotMin = 265.5
ConceptDepotMode = 354
ConceptDepotMax = 442.5
ConceptBaseFixedCost = 1882.188
ConceptLaunchFixedCost = 9.887
ConceptLandingFixedCost = 6.965
ConceptProcessingFixedCost = 12.998
COnceptIntegrationFixedCost = 4.959

sub run
        Set AM = CreateObject( "Arena.Application" )
        AM.Visible = True
        AM.Models.Open ( wrapper.directory & "\RLVSim_ver1.doe")
        Set Siman = AM.ActiveModel.Siman

        ' Turn off Excel output and userform trigger
        AM.ActiveModel.Modules(129).Data( "Initial Value") = 0
        AM.ActiveModel.Modules(73).Data( "Initial value") = 0

if ConceptChoice=1 then
                AM.ActiveModel.Modules(126).Data( "Max Batches" ) = ConceptFleetSize
                if ConceptStageCount=2 then
                        AM.ActiveModel.Modules(9).Data( "Max Batches" ) = ConceptFleetSize
                        AM.ActiveModel.Modules(17).Data( "Batch Size" ) = 2
                else
                        AM.ActiveModel.Modules(9).Data( "Max Batches" ) = 0
                        AM.ActiveModel.Modules(17).Data( "Batch Size" ) = 1
                end if
                AM.ActiveModel.Modules(106).Data( "Initial Value" ) = ConceptDepotTrigger
                AM.ActiveModel.Modules(90).Data( "Value" ) = ConceptMissionsPerDepot
                AM.ActiveModel.Modules(53).Data( "Percent True" ) = ConceptReliability*100
                AM.ActiveModel.Modules(67).Data( "Initial Value" ) = ConceptLOVReplacement
                AM.ActiveModel.Modules(117).Data( "DelayType" ) = ConceptReplacementTime
                AM.ActiveModel.Modules(68).Data( "DelayType" ) = ConceptReplacementTime
                AM.ActiveModel.Modules(27).Data( "Initial Value" ) = ConceptProcessingCPD
                AM.ActiveModel.Modules(28).Data( "Initial Value" ) = ConceptIntegrationCPD
                AM.ActiveModel.Modules(29).Data( "Initial Value" ) = ConceptRunwayCPD
                AM.ActiveModel.Modules(96).Data( "Initial Value" ) = ConceptMaintenanceCost
                AM.ActiveModel.Modules(78).Data( "Initial Value" ) = ConceptCostIncurred
                AM.ActiveModel.Modules(79).Data( "Initial Value" ) = ConceptQueueCostFactor
                AM.ActiveModel.Modules(20).Data( "Capacity" ) = COnceptOPFCapacity
                AM.ActiveModel.Modules(48).Data( "Capacity" ) = ConceptBPFCapacity
                AM.ActiveModel.Modules(31).Data( "Capacity" ) = ConceptIntCapacity
                AM.ActiveModel.Modules(38).Data( "Capacity" ) = ConceptRunwayCapacity
                AM.ActiveModel.Modules(1).Data( "Min" ) = ConceptOPFTimeMin
                AM.ActiveModel.Modules(1).Data( "Value" ) = ConceptOPFTimeMode
                AM.ActiveModel.Modules(1).Data( "Max" ) = ConceptOPFTimeMax
                AM.ActiveModel.Modules(2).Data( "Min" ) = ConceptIntMin
                AM.ActiveModel.Modules(2).Data( "Value" ) = ConceptIntMode
                AM.ActiveModel.Modules(2).Data( "Max" ) = ConceptIntMax
                AM.ActiveModel.Modules(3).Data( "Min" ) = ConceptOrbiterMissionMin
                AM.ActiveModel.Modules(3).Data( "Value" ) = ConceptOrbiterMissionMode
                AM.ActiveModel.Modules(3).Data( "Max" ) = ConceptOrbiterMissionMax
                AM.ActiveModel.Modules(10).Data( "Min" ) = ConceptBPFTimeMin
                AM.ActiveModel.Modules(10).Data( "Value" ) = ConceptBPFTimeMode
                AM.ActiveModel.Modules(10).Data( "Max" ) = ConceptBPFTimeMax
                AM.ActiveModel.Modules(18).Data( "Min" ) = ConceptBoosterMissionMin
                AM.ActiveModel.Modules(18).Data( "Value" ) = ConceptBoosterMissionMode
                AM.ActiveModel.Modules(18).Data( "Max" ) = ConceptBoosterMissionMax
                AM.ActiveModel.Modules(37).Data( "Min" ) = ConceptRunwayMin
                AM.ActiveModel.Modules(37).Data( "Value" ) = ConceptRunwayMode
                AM.ActiveModel.Modules(37).Data( "Max" ) = ConceptRunwayMax
                AM.ActiveModel.Modules(95).Data( "Min" ) = ConceptDepotMin
                AM.ActiveModel.Modules(95).Data( "Value" ) = ConceptDepotMode
                AM.ActiveModel.Modules(95).Data( "Max" ) = ConceptDepotMax
                AM.ActiveModel.Modules(139).Data("Initial Value") = ConceptBaseFixedCost+ConceptLandingFixedCost+
                        ConceptLaunchFixedCost*ConceptRunwayCapacity+ConceptProcessingFixedCost*
                        ConceptOPFCapacity+ConceptIntegrationFixedCost*ConceptIntCapacity
end if
```

```
if ConceptChoice=2 then
                AM.ActiveModel.Modules(126).Data( "Max Batches" ) = ConceptFleetSize
                AM.ActiveModel.Modules(9).Data( "Max Batches" ) = 0
                AM.ActiveModel.Modules(17).Data( "Batch Size" ) = 1
                AM.ActiveModel.Modules(106).Data( "Initial Value" ) = 1
                AM.ActiveModel.Modules(90).Data( "Value" ) = 8
                ' AM.ActiveModel.Modules(53).Data( "Percent True" ) = 99.8
                AM.ActiveModel.Modules(53).Data( "percent True" ) = ConceptReliability
                AM.ActiveModel.Modules(67).Data( "Initial Value" ) = 0
                AM.ActiveModel.Modules(117).Data( "DelayType" ) = 365
                AM.ActiveModel.Modules(68).Data( "DelayType" ) = 365
                AM.ActiveModel.Modules(27).Data( "Initial Value" ) = 1.008731
                AM.ActiveModel.Modules(28).Data( "Initial Value" ) = 2.106812
                AM.ActiveModel.Modules(29).Data( "Initial Value" ) = 1.571859
                AM.ActiveModel.Modules(96).Data( "Initial Value" ) = 108
                AM.ActiveModel.Modules(78).Data( "Initial Value" ) = 0
                AM.ActiveModel.Modules(79).Data( "Initial Value" ) = 0
                ' AM.ActiveModel.Modules(20).Data( "Capacity" ) = 2
                ' AM.ActiveModel.Modules(48).Data( "Capacity" ) = 0
                ' AM.ActiveModel.Modules(31).Data( "Capacity" ) = 2
                ' AM.ActiveModel.Modules(38).Data( "Capacity" ) = 2
                AM.ActiveModel.Modules(20).Data( "Capacity" ) = COnceptOPFCapacity
                AM.ActiveModel.Modules(48).Data( "Capacity" ) = ConceptBPFCapacity
                AM.ActiveModel.Modules(31).Data( "Capacity" ) = ConceptIntCapacity
                AM.ActiveModel.Modules(38).Data( "Capacity" ) = ConceptRunwayCapacity
                AM.ActiveModel.Modules(1).Data( "Min" ) = 46.499325
                AM.ActiveModel.Modules(1).Data( "Value" ) = 61.9991
                AM.ActiveModel.Modules(1).Data( "Max" ) = 77.498875
                AM.ActiveModel.Modules(2).Data( "Min" ) = 4.20435
                AM.ActiveModel.Modules(2).Data( "Value" ) = 5.6058
                AM.ActiveModel.Modules(2).Data( "Max" ) = 7.00725
                AM.ActiveModel.Modules(3).Data( "Min" ) = 7.5
                AM.ActiveModel.Modules(3).Data( "Value" ) = 10
                AM.ActiveModel.Modules(3).Data( "Max" ) = 12.5
                AM.ActiveModel.Modules(10).Data( "Min" ) = 0
                AM.ActiveModel.Modules(10).Data( "Value" ) = 0
                AM.ActiveModel.Modules(10).Data( "Max" ) = 0
                AM.ActiveModel.Modules(18).Data( "Min" ) = 0
                AM.ActiveModel.Modules(18).Data( "Value" ) = 0
                AM.ActiveModel.Modules(18).Data( "Max" ) = 0
                AM.ActiveModel.Modules(37).Data( "Min" ) = 17.87565
                AM.ActiveModel.Modules(37).Data( "Value" ) = 23.8342
                AM.ActiveModel.Modules(37).Data( "Max" ) = 29.79275
                AM.ActiveModel.Modules(95).Data( "Min" ) = 265.5
                AM.ActiveModel.Modules(95).Data( "Value" ) = 354
                AM.ActiveModel.Modules(95).Data( "Max" ) = 442.5
                AM.ActiveModel.Modules(139).Data("Initial Value") = 1882.188+6.965+9.887*ConceptRunwayCapacity+12.998*ConceptOPFCapacity
                        +4.959*ConceptIntCapacity
end if
if ConceptChoice=3 then
                AM.ActiveModel.Modules(126).Data( "Max Batches" ) = ConceptFleetSize
                AM.ActiveModel.Modules(9).Data( "Max Batches" ) = ConceptFleetSize
                AM.ActiveModel.Modules(17).Data( "Batch Size" ) = 2
                AM.ActiveModel.Modules(106).Data( "Initial Value" ) = 1
                AM.ActiveModel.Modules(90).Data( "Value" ) = 8
                AM.ActiveModel.Modules(53).Data( "Percent True" ) = 99.95
                AM.ActiveModel.Modules(67).Data( "Initial Value" ) = 0
                AM.ActiveModel.Modules(117).Data( "DelayType" ) = 180
                AM.ActiveModel.Modules(68).Data( "DelayType" ) = 180
                AM.ActiveModel.Modules(27).Data( "Initial Value" ) = 0.77815
                AM.ActiveModel.Modules(28).Data( "Initial Value" ) = 0
                AM.ActiveModel.Modules(29).Data( "Initial Value" ) = 0.54264
                AM.ActiveModel.Modules(96).Data( "Initial Value" ) = 11
                AM.ActiveModel.Modules(78).Data( "Initial Value" ) = 0
                AM.ActiveModel.Modules(79).Data( "Initial Value" ) = 0
                AM.ActiveModel.Modules(20).Data( "Capacity" ) = ConceptOPFCapacity
                AM.ActiveModel.Modules(48).Data( "Capacity" ) = ConceptBPFCapacity
                AM.ActiveModel.Modules(31).Data( "Capacity" ) = 20
                AM.ActiveModel.Modules(38).Data( "Capacity" ) = ConceptRunwayCapacity
                AM.ActiveModel.Modules(1).Data( "Min" ) = 5.0706825
                AM.ActiveModel.Modules(1).Data( "Value" ) = 6.76091
                AM.ActiveModel.Modules(1).Data( "Max" ) = 8.4511375
                AM.ActiveModel.Modules(2).Data( "Min" ) = 0
                AM.ActiveModel.Modules(2).Data( "Value" ) = 0
                AM.ActiveModel.Modules(2).Data( "Max" ) = 0
                AM.ActiveModel.Modules(3).Data( "Min" ) = 1.5
                AM.ActiveModel.Modules(3).Data( "Value" ) = 2
                AM.ActiveModel.Modules(3).Data( "Max" ) = 2.5
                AM.ActiveModel.Modules(10).Data( "Min" ) = 5.0706825
                AM.ActiveModel.Modules(10).Data( "Value" ) = 6.76091
                AM.ActiveModel.Modules(10).Data( "Max" ) = 8.4511375
                AM.ActiveModel.Modules(18).Data( "Min" ) = 0.375
                AM.ActiveModel.Modules(18).Data( "Value" ) = 0.5
                AM.ActiveModel.Modules(18).Data( "Max" ) = 0.625
                AM.ActiveModel.Modules(37).Data( "Min" ) = 1.7049675
                AM.ActiveModel.Modules(37).Data( "Value" ) = 2.27369
                AM.ActiveModel.Modules(37).Data( "Max" ) = 2.8421125
                AM.ActiveModel.Modules(95).Data( "Min" ) = 26.232375
                AM.ActiveModel.Modules(95).Data( "Value" ) = 34.9765
                AM.ActiveModel.Modules(95).Data( "Max" ) = 43.720625
```

```
                AM.ActiveModel.Modules(139).Data("Initial Value") =  75.051+0.455+0.599*ConceptRunwayCapacity+1.930*ConceptOPFCapacity
        end if

        ' Run Model
        AM.ActiveModel.FastForward

        LaunchCOunt =           Siman.TallyAverage(3)
        AverageTurnaround =     Siman.TallyAverage(4)
        AverageLaunchCost =     Siman.TallyAverage(5)
        TotalCost  =            Siman.TallyAverage(2)
        LOVCount =              Siman.TallyAverage(1)
        LaunchesPerYear =       LaunchCount/20
        AverageCostPerYear =    TotalCost/20
        OpsCostPerFlight =      TotalCost/LaunchCount

        AM.ActiveModel.end
        AM.ActiveModel.Save
        AM.ActiveModel.Close
        AM.Quit
end sub
```