

IAC-09.D2.4.5

FORMULATING DISCRETE EVENT SIMULATION FOR DEVELOPMENT AND ACQUISITION COST ANALYSIS OF REUSABLE LAUNCH VEHICLES

A.C. Charania*

SpaceWorks Commercial, a division of SpaceWorks Engineering, Inc. (SEI), United States of America (USA)
ac@sei.aero

Michael Kelly[†]

SpaceWorks Engineering, a division of SpaceWorks Engineering, Inc. (SEI), United States of America (USA)
michael.kelly@sei.aero

John R. Olds[‡]

SpaceWorks Engineering, a division of SpaceWorks Engineering, Inc. (SEI), United States of America (USA)
john.olds@sei.aero

Dominic J. DePasquale[§]

SpaceWorks Engineering, a division of SpaceWorks Engineering, Inc. (SEI), United States of America (USA)
dominic.depasquale@sei.aero

Ronald P. Menich^{<¶}

SpaceWorks Engineering, a division of SpaceWorks Engineering, Inc. (SEI), United States of America (USA)
ron.menich@sei.aero

ABSTRACT

Future reusable launch vehicles are considered to have potential benefits compared to existing and/or expendable systems. These benefits are likely to arise from reductions in turnaround time and recurring operations costs in labor and materials, potentially at the expense of additional development and acquisition cost. SpaceWorks Engineering Inc. (SEI) has developed, through research and quantitative modeling, a discrete event simulation framework referred to as Descartes-Hyperport to predict turnaround time, recurring ground operations cost, and other operations metrics for future launch vehicles. This type of modeling goes beyond traditional historical modeling efforts by focusing on subsystem-level processes and how the resources required for these processes are utilized. The models are implemented in the discrete-event simulation software Arena. Inputs for the model include descriptions of the basic vehicle concept of operations, mission models, and performance information. Outputs presented include estimated times and costs. This paper presents some of the initial work by the authors in conceptualizing how a discrete-event simulation framework could also be used for development and acquisition cost modeling.

NOMENCLATURE

DDT&E Design, Development, Test, &

DES	Evaluation
KSC	Discrete Event Simulation
I-RaCM	Kennedy Space Center
IVHM	Integrated Risk and Cost Model
LaRC	Integrated Vehicle Health Management
LCC	Langley Research Center
LOM	Life Cycle Cost
LOV	Loss of Mission
NRA	Loss of Vehicle
OMS	NASA Research Announcement
RCS	Orbital Maneuvering System
	Reaction Control System

* - *President, Senior Member AIAA.*

[†] - *Operations Engineer, Member AIAA.*

[‡] - *Principal Engineer, Associate Fellow AIAA.*

[§] - *System Engineer, Member AIAA.*

^{<¶} - *Space Scenario Analyst.*

SEI	SpaceWorks Engineering, Inc.
STS	Space Transportation System
TFU	Theoretical First Unit
TPS	Thermal Protection System
VBA	Visual Basic for Applications

INTRODUCTION

In the not-too-distant future, it is extremely likely that the aerospace industry will design and build another generation of reusable space launch vehicles. As the amount of space-bound traffic continues to increase over the next few decades, reusable vehicles have the potential to offer significant cost savings over existing or new expendable launch systems. While reusability offers obvious advantages in total fleet acquisition costs for any extended program, the key to maintaining this advantage lies in designing vehicles that are cheap and efficient to operate. Vehicle turnaround from landing to launch must be performed quickly enough to support increased demand for launches, and be inexpensive enough to justify potentially higher development and per-vehicle production costs. Developing accurate estimates of these turnaround times and recurring costs has historically proven to be difficult for the space systems analysis community. This study effort at SpaceWorks Engineering, Inc. (SEI) has developed new tools for estimating this information for future reusable launch vehicles, and is currently using the same modeling philosophies to build tools for estimating Design, Development, Test and Evaluation (DDT&E) and Theoretical First Unit (TFU) costs. When complete, these tools can together be applied to estimate Life Cycle Cost (LCC), allowing program managers and research teams a more complete and accurate picture of the financial consequences of their vehicle design decisions.

Operations Analysis

The ground operations models in use when the Space Shuttle (or Space Transportation System, STS) was developed grossly underestimated the ground operations challenges, and therefore the resources involved, in preparing the shuttle for flight. Official estimates from 1976 included the statement: “The Space Shuttle Orbiter is designed for a 2-week ground turnaround, from landing to relaunch. About 160 hours of actual work will be required.”¹ The reality, of course, has involved orders of magnitude more work, and a much more complex operations

environment (see Fig. 1). From 1990 to 1997, the average time spent in the Orbiter Processing Facility (OPF) was 88 days.² Accounting for vehicle stage integration, launchpad activities, and periodic depot maintenance, it is easy to see why the shuttles spend so much more time on the ground than was originally planned.



Figure 1. Expectation and Reality of Space Shuttle Turnaround Operations (Orbiter Surrounded by Platforms and Equipment in Right Image)

In the last few decades, teams working at various NASA centers and other research groups in the aerospace industry have addressed the challenge of ground operations analysis in various ways. LLEGO, and its predecessors AAT and AATe, are spreadsheet-based models built primarily at Kennedy Space Center (KSC) using extensive STS data as the basis for most of the formulas. RMAT, maintained at Langley Research Center (LaRC), is a database model built with a similar approach. SOVOCS, REWARD, COMET/OCM are some of the many other models currently available, but all of these (along with LLEGO and RMAT) share a critical limitation. They are all entirely analogy-based. They attempt to fit a curve to historical data points (often just the shuttle data), then allow a user to input whether a system is more similar to one vehicle or another, and sum a large quantity of component results together. While this is not an inherently flawed approach, it does limit the range of operational philosophies that can be modeled. It is unable to account for interaction of processes and resources, as larger fleet sizes are processed in parallel. As new technologies are developed and new procedures adopted, a new modeling paradigm is needed to accurately describe ground operations.

Discrete Event Simulation

Discrete Event Simulation (DES), also known as the event scheduling approach, is a technique developed in the industrial and systems engineering community that has been applied in industries as varied as

healthcare, banking, manufacturing, shipping, and commercial aviation. The general idea is that any complex set of processes can be described in terms of events occurring at discrete points in time. A non-instantaneous event is re-defined by its start point and end point, which are both instantaneous. A DES model keeps track of all the events that are scheduled to occur. As time moves forward, when an event is reached on the schedule, its effects on the system (e.g. new resources becoming available, an inspection being completed, etc.) are recorded, sometimes new events are added to the schedule, and the simulation moves on to the next event. This is very computationally efficient, since it only requires a single event to be processed at a time, and it enables very complex systems to be analyzed by breaking them down into discrete events and placing all the events on a single schedule. DES can be a powerful tool for many kinds of systems analysis, and is ideally suited to the problem of ground operations.

Basic DES models can be built in spreadsheets, but larger models generally employ DES-specific software. Rockwell Automation's Arena software, built on the DES language SIMAN, is one of the industry leaders and was selected by SEI for this study. Arena's Basic Edition gives users a 'Basic Process Panel,' which allows users to build models with 8 types of modules, or 'blocks': create, dispose, process, decide, batch, separate, assign, and record. It also gives spreadsheet-based control over resources, entities (and their attributes) and variables. These model components will be described as they are encountered in the model description in section III below. The blocks are arranged in a graphical user interface and linked together to represent actual sequences of processes, logic-based decisions, etc. The end result bears a strong resemblance to a flow-chart. Additionally, Arena allows the model to be animated while running, giving users a chance to observe the flow of entities through the system and, if desired, live-updating statistics related to system performance.³

DESCARTES MODELING FRAMEWORK OVERVIEW

SpaceWorks Engineering conceived of the idea to develop DES-based models for analyzing turnaround time, operations costs, DDT&E costs, and TFU costs. After being granted funding through a NASA Research Announcement (NRA) (see Acknowledgments for details), SEI began work in early 2008. In addition to the selection of Arena

Basic Edition as the central DES engine, several other modeling conventions were laid out. Collectively, these decisions defined what was named the Descartes modeling framework. Over the three years of funding, the SEI team would use the Descartes framework in two phases. Phase I consisted of completely defining the framework structure and building a model for estimating turnaround time and recurring costs. Phase II focuses on a second model for estimating for DDT&E and TFU costs.

The structure of a Descartes model involves an Arena (.doe) model file along with several MS Excel 2003 (.xls) files. One Excel file has a user input worksheet, which is filled out with the vehicle design description and several modeling parameters. This file contains another worksheet in which the user inputs are converted into a number of variable arrays, arranged for input into the Arena model. Supporting this conversion is a second .xls file containing a historical database of relevant data, collected to suit the needs of that particular model. Excel's native Visual Basic for Applications (VBA) then opens the .doe file, and Arena's native VBA imports the values from the worksheet into Arena variable arrays. The model runs for a given number of replications (to increase the statistical validity of the outputs). After each replication, data is exported to a new worksheet in the output .xls file. After all runs are completed, this .xls file calculates averages and confidence intervals of the various metrics. This general file structure and data flow-path is followed for all Descartes models, allowing for consistent and reliable outputs that can be jointly utilized for more complete program analysis. To this end, it is a long-term goal for each Descartes model to be integrated into Phoenix Integration's ModelCenter tool, which allows the inputs and outputs from various design and analysis tools to be fed into each other. This simplifies not only initial analysis efforts, but also enables parametric sweeps and trade studies to be performed more efficiently. More specifically, we intend to make both Descartes models part of SEI's Integrated Risk and Cost Model (I-RaCM), a suite of cost and risk analysis tools integrated through ModelCenter.⁴

DESCARTES-HYPERPORT

The first complete Descartes model built was Descartes-Hyperport. Hyperport is designed to estimate turnaround time and recurring costs for reusable space launch vehicles. The model accepts as inputs a description of the vehicle along with the

program's operation parameters. It can handle designs ranging from two stage fully reusable rocket-based combined cycle vehicles to single stage reusable rockets to reusable turbine-based combined cycle boosters with expendable upper stages, or any permutation along these lines. Version 1.0 was completed in early January, 2009, while the latest version, 1.2, was released in early July, 2009. This marked the final completion of Phase I of the NRA contract. All descriptions in the remainder of this paper refer to version 1.2. As explained above, Hyperport consists of a set of several files. Hyperport.doe, the Arena model file, contains approximately 700 modules that, between them, describe a standardized operations flow through a notional turnaround maintenance facility. Hyperport.xls is the input workbook, with the Descartes-standard user input sheet, followed by the Arena input sheet. This file also contains an extra sheet used to help the user come up with realistic resource inputs for their first few simulation runs. DescartesData.xls is the supporting database file. A new output file is created each run with a name input by the user. All of these files, taken together, constitute the Hyperport model.

Hyperport Arena Model

As stated, the .doe Arena model file is the DES engine at the heart of any Descartes model, including Hyperport. It is possible, under certain conditions, to run the Arena file independently of the .xls files, but it is generally accessed through Hyperport.xls. However, in this subsection the entity and process flows are described within the context of the standalone Arena file.

In this model, the modules are divided amongst 11 'submodels.' Submodels appear similar to process blocks when viewing an animated Arena model, but function more like folders. While selecting regular blocks pulls up a set of data fields to describe how they operate, double-clicking on a submodel opens a window, which appears as another model screen, containing sets of blocks linked together. This modeling convenience allows for logical groupings to be visually separated from the top-level flow. Hyperport's submodels correspond to major steps in the turnaround process: entity creation, landing, demate, booster stage (S1) turnaround, upper stage (S2) turnaround, stage integration, preflight activities, the mission, and 4 depot submodels: S1 airframe, S1 propulsion, S2 airframe, and S2 propulsion. In Fig. 2 below, most submodels are indicated by an icon representing what takes place therein.

In a particular Arena replication, the first submodel accessed is Creation. Since this is not used frequently, it is grouped with the landing submodel behind the leftmost orange icon. Entities leaving Creation move into their respective "S1" or "S2 Turnaround" submodels, where they are individually readied for flight. From here they both (if it is a two-stage vehicle) move to Stage Integration. Once integrated, the launch vehicle entity is fueled, loaded with any late payload, and positioned for launch in Preflight. If model animation is turned on, at this point the vehicle entity can be seen moving to the runway, "taking off" just above the runway, then "landing" at the other end. The actual mission events, including any loss of vehicle or loss of mission events, are in the Mission submodel, hidden along with Creation behind the Landing icon. The Landing submodel represents any crew egress, other unloading, and engine safing. If some kind of failure resulted in two stages landing together, they are routed to the Demate model for separation. Otherwise, each stage is sent to its respective Turnaround submodel. When entering Turnaround, if it is determined that the airframe or any of its engines require depot maintenance, they are detoured to the appropriate Depot submodel before returning to the regular process flow.

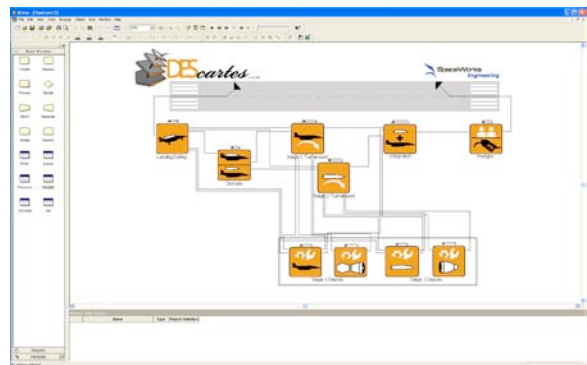


Figure 2. Screenshot of Hyperport.doe

In all of these submodels, most of the processes are of the type 'Seize Delay Release.' This means that a defined set of resources must be available for the process to start, and that they are held until the process is completed, then released to work on the next process. There are resources representing both the various facilities as well as ten types of labor, representing technicians with different specializations. Each submodel's process blocks are programmed to refer to a variable array defining not only what the estimated man-hours are, but also what

quantities of specialists and generalists are needed to complete the work. These variable arrays are imported from Excel (explained further in section III.B) but for man-hours contain only initial estimates. With some exceptions for specific processes, there are three adjustments made to these estimates to determine the length of the delay for each process. First, a random number is taken from a triangular distribution defined symmetrically around the estimate, with a width defined by the user. Second, a Crawford learning curve is applied based on the number of times that process has been performed. The learning rate is also a user input. Finally, the time required for each process is calculated by dividing the adjusted man-hours of work by the quantity of labor working. There are some processes that cannot be done more quickly with more manpower, such as fueling in the preflight submodel, and others that do not experience learning such as taxiing after landing, but this general form holds for most of the approximately 300 process blocks in Hyperport.

Creation

Within the creation submodel, there are create blocks for each of S1 airframe, S1 rockets, S1 turbines, S1 ram/scramjets, and the same pieces for S2. The first airframes (one each of S1 and S2 for fully reusable systems) are created at time 0, with additional creations taking place at intervals taken from an input variable. This is useful to represent programs that will start with a single vehicle, and scale upwards over a period of months or years. At each airframe creation, the appropriate quantity of all included propulsion systems are also created. For example, if a booster contains two rockets and four turbines, at time 0 there will be two S1 rockets created and four S1 turbines created.

The modules immediately following the set of creates are assign blocks. This means that a simulation entity, representing either an airframe or an engine, will visit an assign block as soon as it is created. Each of these tags the entity with a serial number by setting the appropriate entity attribute. For example, an orbiter airframe will be given an attribute called "S2 AF serial number," which will take the next available value. This next available value is stored in a variable array, and incremented by the assign block as soon as the current value has been given to the new entity. This is an example of how a single event in a DES, in this case the entity visiting an assign block, can have multiple sets of instructions which

are all performed by the simulation before it moves to the next scheduled event. Finally, each new entity is given counters for the number of flights remaining before it requires a depot visit, and before it is to be retired. These are stored in variable arrays, with each existing airframe and engine's "flights needed until depot (or retirement)" kept in a single cell of an array, indexed by serial number.

After receiving serial numbers, the propulsion entities move to a batch block, where they will wait until one complete set of engines has been created. Using the previous example, once two S1 rockets reach the batch block they will be temporarily combined into an "S1 rocket system" entity, and four S1 turbines will form an "S1 turbine system" entity. These temporary batches are used extensively to allow a number of entities to move through the model flow as one while maintaining their original attributes. After engines are batched they move to an additional batch point where the airframe accepts one of each kind of engine system that it needs, and these form either a "S1 stage" or "S2 stage" entity. This stage maintains the tail number (serial number) of the airframe. This complete stage entity exits the creation submodel and moves on to its respective stage turnaround submodel. Before covering the turnaround submodels, we will describe the other path entities take to arrive there.

Mission

Once a vehicle departs from the Preflight submodel, it enters into its Mission. While this may seem to make Mission the last submodel, it goes along with Creation in determining when and in what condition stage entities arrive for turnaround. This submodel, seen in Fig. 3, models the various potential outcomes of a mission. The determination of outcome is based on a series of logical gates and reliability-driven random decisions. For example, in a two-stage vehicle, there could be a loss of vehicle (LOV) event during mated flight, which would result in both stages being destroyed. Alternatively, the stages could separate and the orbiter proceeds on to a successful mission, while the booster suffers some sort of problem on its return flight that will force an emergency landing. The complete set of possible outcomes from a single mission are:

1. Regular landings. For S1 and/or S2, if all random failure events are avoided, the stage will proceed to a regular landing.
2. Loss of mission (LOM). For S1, S2, or the mated vehicle, there could be an LOM failure that will result in an emergency landing.
3. Loss of vehicle. For S1, S2, or the mated vehicle, there could be an LOV failure. If this happens, the entities for the 'lost' stages are duplicated. If the user has selected to replace lost vehicles, one copy will wait for the given amount of time before having its tail number and engine serial numbers changed to represent a new vehicle, then passing into stage turnaround at the same point newly-created vehicles enter that submodel. The second copy seizes a resource called "Launch Clearance" for the duration of a simulated 'stand-down' period. The affect of this is to assume that after an LOV event, there will be some investigation that prevents any new flights from launching for a significant period of time. Without the launch clearance, any vehicles in process will be held up in preflight until the stand-down has concluded.

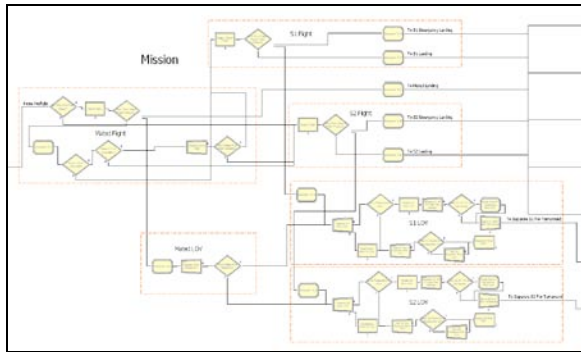


Figure 3. Mission Submodel

Landing and Demate

The Landing submodel has five parallel flows, each structured similarly. There is one flow for each of S1 and S2 regular landings and emergency landings, and a fifth for mated emergency landings. The sequence of processes is close to identical in all five paths, but the times and resources required are different. In an emergency landing (mated or unmated), it is assumed that the loss of mission event increased the likelihood of some sort of damage taking place to the vehicle. For this reason, an emergency landing will result in more work being required to complete comparable tasks. A mated emergency landing naturally requires

more the total amount of work usually required for each individual stage, plus some added time to account for difficulties in access.

In any of these cases, the stage (or stages) arrive from the mission model and their first visit is to an assign block, labeled as a 'checkpoint.' At the beginning (and end) of all top-level submodels, there are checkpoints set up which record the entity's arrival (departure) into a large variable array called the 'Event History.' Each row of this five-column array records the specific event that has occurred, the entity type passing through, the entity serial/tail number, and the current simulation time. Since every checkpoint reports all of its visitors into this array, the end result is an ordered list of how entities moved through the Arena model. This data is processed in the output workbook to generate various time interval metrics.

After the pre-landing checkpoint, a vehicle on any of the five landing flows will go through landing and taxi maneuvers. Here it is generally assumed that the vehicles are capable of a horizontal runway landing, and will land at their primary maintenance facility. If either of these assumptions does not hold for a vehicle being studied, some hard-coded parameters can be adjusted to reflect the actual operations flow. If there are any crew on board, they egress from the vehicle immediately after taxi. For visual simplicity, the next several processes are grouped into submodels within each landing flow. Collectively they encompass engine safing, and include processes to purge tanks, install engine covers, and any other steps that are generally taken outdoors, on the flight line, before the vehicle enters an enclosed space. There are a set of separate blocks that allow each propulsion system to be processed individually according to its needs, then a set of batch points to recombine the entities into a whole stage (or mated pair of stages).

At this point, the stage is routed according to what condition it is in. A routine landing is sent to its respective turnaround facility. Due again to the assumption that an emergency landing is the result of some sort of major failure that will require additional attention, an emergency individual stage landing is tagged to be sent off to the depot, for an "unscheduled" depot visit. A mated emergency landing is sent to the Demate facility submodel, where the stages are separated and then sent to their unscheduled depot visits. The Demate facility itself is

modeled simply. Consisting of a single demate process surrounded by some logical routing blocks.

Turnaround Submodels

The heart of the Arena model, and the most complex submodels, are the S1 and S2 Turnaround submodels. Since upper stages contain a few more subsystems that are modeled in Hyperport, the S2 Turnaround submodel is slightly more complex and will be used as the sample case in this section. Note that the only differences between the two are the subsystems from S2 that are not included in S1.

The turnaround submodels are broken up into several different areas, indicated by the orange boxes drawn around sets of blocks as seen in Fig. 4. Entities enter from creation, landing, demate, or as replacements from retirements or LOVs on the left. If arriving from landing or demate, they are first checked to see if they are due for a depot visit by checking if the appropriate ‘flights until depot’ variable has reached 0. If the vehicle is found to be ready for depot but the depot already has an airframe of this type in process, the depot visit is postponed until the next flight to improve airframe availability as discussed above. If an airframe is to be sent to the depot, there is some preparation required. First the engines (which may be on a different depot schedule and, in any case, are sent to their own depot facilities) must be removed. Rockets and turbines, if present, are always removed and sent back into the regular turnaround flow. High-speed air-breathing engines such as ramjets or scramjets are removed only if they are not integrated into the airframe. After these steps, the airframe is free to pass on to its depot.

All other arriving stages, as well as those who pass through the depot check, are sent to a checkpoint, they seize one unit of a turnaround facility resource, then they are split into airframe and propulsion systems. The main turnaround processes are split into a system of fourteen parallel submodels (eleven for S1), eleven for airframe subsystems (eight for S1) and three for propulsion. These submodels represent maintenance activities for each of thermal protection systems (TPS), power, avionics, environmental controls, hydraulics and actuators, reaction control systems (RCS), mechanical and pyro systems, cockpit and crew cabin, orbital maneuvering systems (OMS, S2 only), thermal controls (S2 only), payload (S2 only), rockets, turbines, and ram/scramjets. All of these are treated as parallel flows, implying that many subsystems will be underway at a time. For the

various engine types, this is straightforward since each type of engine goes to its respective submodel. However, for the airframe subsystems, the airframe entity is duplicated and copies of it are sent to each submodel. These duplicates are re-batched into a single airframe after all subsystem turnarounds are complete.

The airframe subsystem submodels have varying levels of detail, positively correlated with the quantity of work that is usually done on that particular system. For example, the cockpit and crew cabin is broken into three processes (with some additional logical blocks), while TPS involves twenty-one process blocks. Depending on the work needed, these blocks may be in parallel, series, or some combination. TPS maintenance involves several parallel flows, each containing a series of steps needed to inspect and repair a different TPS component. In this specific case, the blocks are generically titled since different materials may require different processes, or a different sequence of processes. Waterproofing is generally required for tiles, but not blankets. Some materials may require wholesale replacement, while others are generally repaired in smaller sections.

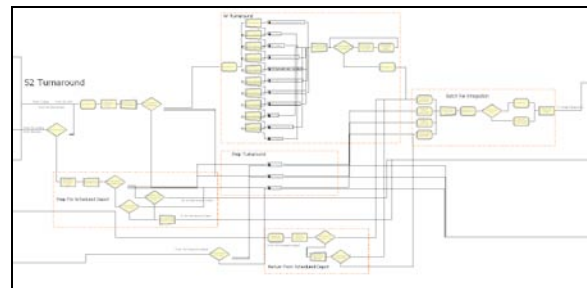


Figure 4. S2 Turnaround Submodel

Eight of the ten different types of labor resources defined in Hyperport are specific to stage turnaround. ‘General Turnaround Staff’ are used for some basic processes such as the cleaning and inspection of the payload bay, while ‘Electrical Specialists’ are needed for power and avionics work, TPS specialists are needed for TPS work, etc. In many cases these specialist labor types may also be assisted by the general technicians. In most practical scenarios, there are not enough labor available to work on all processes in parallel, so technicians will move around from subsystem to subsystem until all the work is completed.

Propulsion Turnaround

The most complex of the subsystem turnaround models are those representing engine maintenance. Fig. 5 shows the S2 turbine maintenance process flow. There are four general areas in this (and the other propulsion) submodel. First, on the left, are the initial inspections. The primary maintenance processes and decisions are in the middle. Processes representing engine removal and replacement are in the lower portion of the screen. Final inspections are to the right. While turbines are pictured in Fig. 5, we will discuss the model in terms of all three propulsion types.

The diamond-shaped block at the far left is a decision based on whether the stage contains the relevant propulsion system. Only an S2 turbine system entity can pass this block, any others would be routed around the model directly to the exit. After this decision, and a checkpoint, the propulsion system entity is divided back into its individual engines. The engine entities are duplicated so that one copy can be inspected physically, while the other represents reading the Integrated Vehicle Health Management (IVHM) reports. There is an inverse relationship between the quality of IVHM systems (a user input) and the amount of time needed for physical inspections. After the two copies of the engine are recombined and it is decided whether the engine requires basic work, major work, or depot maintenance. In real life this decision would be based on the results of the inspections, but in Hyperport it is based on random chance, with probabilities based on engine reliability.

If only basic work is needed, the engine entity travels along the top-most flowpath through a sequence of basic processes (in Fig. 5 it is a single process). If depot work is needed it travels along the bottom flowpath representing engine removal. If the user has chosen to assume that there is an initial supply of spare engines waiting at the turnaround facility, one of them is used to replace the depot-bound engine. If not, the engine is re-attached when it returns from the depot. The middle path represents major, non-depot level work. This is the area where the three engine types differ the most, since the specific maintenance they may require differs so much. Based on previous work within the author's organization, if rockets or ram/scramjets need major work, that work will consume enough time that it is worth the effort to replace the engines and work on them out of the critical path, so those entities go through the

remove/replace process just as if they were being sent to depot, except that the removed engine entity remains in the turnaround submodel. Turbine engines have major work performed while still attached to the airframe since this work can generally be performed quickly enough to avoid seriously affecting the whole vehicle's turnaround time.

It should be noted that while DES does not lend itself to processes being interrupted and restarted, Hyperport is built to accommodate the fact that engine removals may be delicate enough procedures (or may involve hazardous enough materials) that all other activity needs to be halted for a portion of the operation. The total count of engine removals and replacements during a single turnaround cycle is therefore tracked, along with the amount of time those removals take. After the airframe subsystem entity copies are reunified, but before the airframe is considered ready to batch with its engine entities, there is a set of blocks that cause airframe delays in accordance with the amount of time that would have been lost for those engines. While this could result in some lower-level subsystems being repaired more quickly in the simulation than in reality, it induces an accurate lag for the whole airframe. After accounting for these delays, the airframe is routed to the 'batch for integration' area (labeled in Fig. 4).

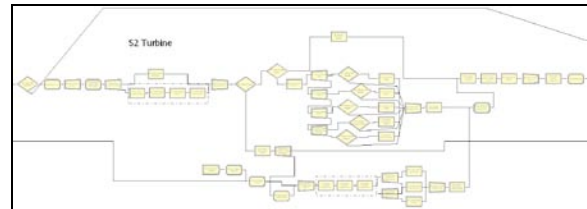


Figure 5. S2 Turbine Turnaround Submodel

Whether going through light, major, or depot work, all engine types are brought back to the same path for functional tests, installation of thermal covers, removal of engine covers, and a final check. The engines are recombined into an engine system batch entity, and after passing a checkpoint they return to the main turnaround submodel. All included engine systems are batched back together with the airframe entity in 'batch for integration' and exit turnaround for the stage integration submodel.

Stage Integration and Preflight Ops

After going through turnaround, all vehicles enter Stage Integration, which has a flow that mirrors the Demate submodel but with some added logical

complexity. The key difference is that Stage Integration has to be able to handle the various vehicle configurations that Hyperport is built to accept. These include single stage reusable, two-stage fully reusable, reusable booster with expendable orbiter, and expendable booster with reusable orbiter. If the vehicle is a single stage, it obviously needs no stage integration, so it is routed immediately to the submodel exit and on to preflight. For two-stage fully reusable vehicles there are several steps. Each stage (after passing through a checkpoint) is prepared for integration. It is then sent to a batch point where it waits for its partner stage. After batching, a flight number is assigned to the whole vehicle, which then goes through the actual integration process before passing another checkpoint and leaving for preflight. For vehicles with one reusable and one expendable stage, if they were to go to the batch point the wait would be infinite, as only reusable stages can be directly modeled in Hyperport. Instead, after being prepared for integration, they bypass the batch and go directly to the integration process. The accuracy of this no-waiting model depends on the program's ability to deliver the needed expendable stage at the right time which, while beyond the scope of the project to explore in depth, we have no reason to doubt. The integration processes and timelines are generally based on the Vehicle Assembly Building at KSC, the only working reusable launch vehicle integration location.

When entering the preflight activities, the first event, before the checkpoint, is to acquire launch clearance. This is represented by a resource which will be available unless another vehicle is presently in preflight activities, or unless the program is in the midst of a stand-down period, as discussed in the Mission submodel section above. It is also possible, depending on user input, that there is an enforced delay between launches, and launch clearance remains unavailable immediately after a launch for that period. Once cleared to begin launch preparation, the vehicle entity adds on any late access payload and is then fueled. (All tanks are filled in parallel based on known volume flow rates.) Finally, it taxis and takes off. In the case of vertical takeoff, the timing of the 'taxi' process can be used to represent roll-out to a launch pad.

Depots

It is assumed for simulation purposes that a depot-level overhaul maintenance is going to be contracted back to the manufacturer of the system in question.

Because of this, the Hyperport model does not estimate any specific resources or costs for depot processes. (SEI has proprietary methods for estimating these costs if needed for integration into campaign cost models external to Hyperport.) The depot still must be included in the operations model since it affects vehicle availability and therefore the ability of the fleet to meet desired launch rates. The default time setting is for regular scheduled visits to last six months for airframes and three months for engines. Unscheduled visits, those that are the result of a major in-flight failure leading to loss of mission, take 50% longer.

There are four depot submodels in Hyperport: S1 Airframe, S1 Propulsion, S2 Airframe, and S2 Propulsion. All are structured very similarly, with the primary differences only being those extra logic blocks necessary to allow propulsion depot submodels to handle any of their stage's types of engines. Any system, airframe or propulsion, arriving at a depot is immediately checked for retirement. This means that the 'flights until retirement' counter for that system, set back in the creation submodel and decremented every landing, is checked to make sure it is still positive. If the system has reached its design life, there is no need for depot maintenance, and instead the entity is given a new serial number (or tail number in the case of airframes) and sent back to turnaround with reset 'flights until retirement' and 'flights until depot' counters. As with LOV replacements, while this design technically does allow the same entity to exist, and the Arena model's internal statistics cannot tell the difference; all data exported to the output spreadsheets is tagged with the serial number attribute. Changing that number has the effect of granting the 'new' system a fresh life history with no past events associated with it. One important subtlety lies in the fact that a system scheduled for depot that ends up being retired is sent back to turnaround immediately. It is assumed that the program managers will know their system is scheduled to retire and will plan to make its replacement available at the appropriate time. If, on the other hand, it is an unscheduled depot visit in which the retirement occurs, there is a lag before the replacement is made available. It is also possible, in the user input sheet, to decide that vehicles will not be replaced upon retirement at all, in which case retiring entities are simply sent to a dispose block.

If a system passes through the retirement check, it moves on to the appropriate depot process (scheduled or unscheduled). As stated, these processes do not

utilize any labor resources. However, in the case of airframes, they do occupy a single unit of either “S1 Depot Facility” or “S2 Depot Facility.” The purpose of this is not to track costs or total visits, but rather to prevent the simulation from sending multiple airframes of the same type to depot at the same time. While an actual program manager would be capable of more intelligently scheduling his airframe’s depot maintenance to ensure he kept as much of the fleet in the regular flight rotation as possible, this resource usage approximates the managerial decision-making well enough for simulation purposes, in that it limits each airframe depot to working on a single vehicle at a time. Naturally, in the event of an unscheduled depot visit this will result in a queue at the appropriate depot, but this is not an unrealistic outcome either. A real unscheduled depot might actually interrupt and preempt a scheduled depot process, but the affect of this switch on the overall system would be negligible. All depot visits eventually result in the ‘flights until depot’ counter being reset for the airframe or engine, and it being sent back to the turnaround submodel to be readied for its next flight.

Hyperport Input and Output Excel Files

Throughout the last section we have mentioned numbers that are user inputs, and others that are stored in variable arrays. While the Arena .doe model is a very good representation of the typical processes required for reusable launch vehicle maintenance, its value as a simulation is naturally limited to the quality of the numbers that are used to drive all of the decisions and processes. In Hyperport, all of those numbers come from the Excel files, Hyperport.xls and DescartesData.xls.

UserInput Worksheet

The user’s input sheet called, appropriately, UserInput, is found in Hyperport.xls. The sheet, shown in Fig. 6, contains 125 questions about the design of a two-stage vehicle, with a configuration question indicating which stages are to be included. Some questions are numerical, others require ‘Yes’ or ‘No’ inputs, while some have drop-down selection menus. Single stage vehicles are treated as upper stages. To analyze mixed reusable/expendable systems one only needs to enter information on the reusable stage. The first questions for each stage include dry weight, mission length, and quantity of crew. There are questions about the kinds and acreages of TPS, quantities of RCS tanks and

thrusters, quantity of OMS engines, and similar key metrics for other subsystems. There are about ten questions apiece for the rockets, turbines, and ram/scramjets on each stage, including such things as thrust, fuel quantities, and inclusion of IVHM.

In addition to the vehicle description, there are run parameters, resource parameters, and reliability parameters, a total of thirty-nine questions in all. The run parameters section includes questions about how many hours per day and days per year the turnaround facility is operating, the number of replications of the Arena model to perform, and the learning rate. The reliability numbers are primarily LOV and LOM rates, in terms of mean flights between these events, for various stages. The resource parameters include quantities for each of four facility resources and the ten labor resources, as well as the fully emcumbered hourly rates for the labor. When actively performing an analysis using Descartes, it is usually these labor quantities that are varied. One final question, positioned just above the run parameters, allows the user to name their output file. This allows someone to run numerous trials with different input parameters and compare them easily afterwards.

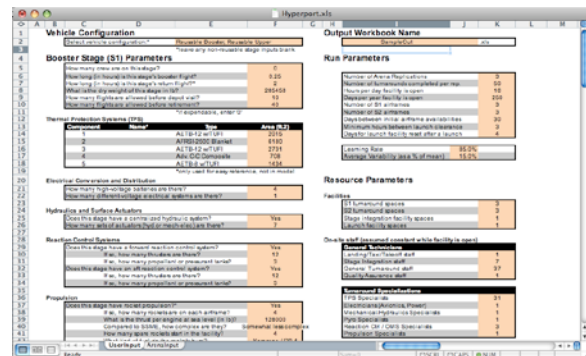


Figure 6. Detail of UserInput Worksheet

To run Hyperport, beyond having the necessary software installed and putting all the files into the same folder, all that is required is to fill out UserInput and run the VBA macro DescartesHyperport from the standard Excel macro dialog box. Alternatively, the keyboard shortcut control-q calls the macro automatically.

DescartesData and the ArenaInput Worksheet

Once the VBA macro has been initiated, the first step is the transfer of data to the worksheet ArenaInput. The run, resource, and reliability parameters are all converted into the formats and units needed by the

Arena model. Some vehicle configuration data such as engine count, which is needed to initiate entities correctly, is also stored with the replication data. After these straightforward conversions, the more complex task of estimating manhours begins. The VBA code opens the file DescartesData.xls, which contains a worksheet for each of the Hyperport subsystems. These worksheets contain the constants and coefficients, derived from historical data, which are used to fill in the VBA formulas calculating required maintenance. In the case of TPS, this is a table with man-hours per square foot for fifteen different materials, broken into up to four sequential processes apiece. Here, the VBA multiplies the relevant cell in the table by the acreage for each material's processes, and stores the results in ArenaInput. For rocket propulsion, DescartesData contains a list of processes and a baseline quantity of manhours for each. These are adjusted based on engine size, complexity, and quality of IVHM before recording in ArenaInput.

to a single process can potentially keep others from getting started early enough, resulting in a net loss of flow time. The individual process weightings have been manually adjusted by the SEI team over the course of several months and various vehicle analyses to make them as efficient as possible.

Running Arena and Processing Outputs

After ArenaInput is filled in, the VBA opens Hyperport.doe, then Excel's VBA hands control over to the VBA in the Arena file. This code scans ArenaInput and copies the values from the spreadsheet into the various places they are needed. As implied above, this includes creation quantities, replication counts, and other run parameter values being inserted into their appropriate places in the model settings. Then each of the submodel three-column arrays from Excel is copied into a three-column variable array in the model file. Before the first replication is initialized, the new .xls output file is created in the folder with the model.

During the replication, checkpoint (assign) blocks are writing rows of data to a variable array called Event History. Once the replication terminates, Arena exports this complete history to a new sheet in the output file and moves on to the next run. In practice, for a two stage reusable vehicle being simulated for 100 flights, the runtime for a single replication is as little as 3-10 seconds, but the copying of data can take 10-30 seconds, as there may be several thousand rows written into the spreadsheet. The Arena model terminates after the last replication and hands control back to Excel.

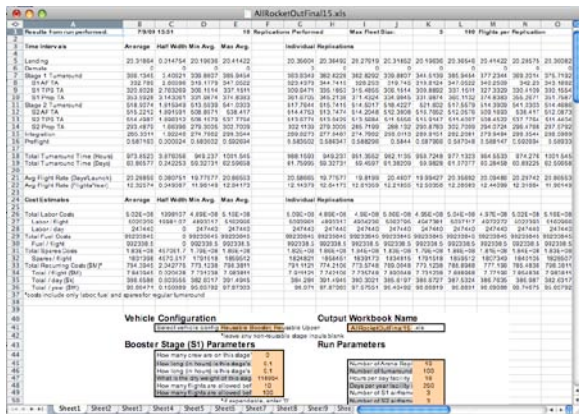


Figure 7. A sample output sheet from Hyperport

Each Arena submodel has its own section of ArenaInput in the form of a three-column variable array, with a row for each process in the submodel. Once the first column has been filled in as described with estimated manhours, the remaining two are filled in with labor resource needs. In most cases, this data takes the form of a quantity of specialists in the second column and a quantity of general technicians in the third. The labor needs for each process are calculated last, since they depend on both the quantities of labor made available, and the expected number of manhours among processes requiring that specialization. Additionally, some high-priority tasks are given slightly higher labor quantities to ensure they are completed more quickly. There is a trade-off inherent here, in that assigning too many technicians

The macro then uses a series of custom-built filtering functions to sort through the event history and pull out the time intervals desired, including average time spent by a vehicle or stage in each facility, average airframe turnaround, average propulsion turnaround, average TPS turnaround, overall average vehicle turnaround, and overall average flight rate. Next, estimates for spares cost and fuel cost are made from the vehicle data for that replication. Labor cost is estimated by calculating the cost per day based on crew size and wage rates, then multiplying by the total time needed for the simulation. Once calculated, all these key outputs from each run are copied into a column of the first worksheet in the file, which has been formatted as an output summary. This sheet, seen in Fig. 7, calculates the averages from all replications, along with minimum and maximum replication outputs and the half-width of a 95%

confidence interval for each output metric. For convenience, the last VBA action is to paste the original UserInput sheet onto the summary output sheet on the rows below the outputs, allowing easy checking of which scenario produced which set of outputs.

Initial Line Personnel Recommendations

The most recent addition to the Hyperport model addresses a need that arose when SEI engineers were utilizing it to analyze a set of two stage vehicles with widely varying maintenance requirements that were both intended to be available for the same number of flights per year. Once the labor quantities were close to correct, it was straightforward to run multiple trials with different configurations, using a heuristic to pinpoint the most efficient solution that met program guidelines. However, narrowing the search range to a reasonable set could take numerous iterations. The time-saving solution to this problem was to build an optimization model that would take advantage of Excel's built in Solver functionality to make intelligent guesses of how many technicians are needed for a vehicle.

Thanks to the ArenaInput sheet, once the model has run at least once, it is not difficult to obtain estimates for the total number of manhours required for a vehicle turnaround. Likewise, this can be broken down into the manhours of each labor specialization required, using knowledge of the processes themselves to allocate that work. If a particular set of tasks such as S1 Power requires 135 manhours per turnaround and the target flight rate is 12 flights/year, then it is known that a total of at least 1620 ($=135*12$) manhours must be provided to complete the processing. The S1 Power work requires one or more electrical specialists and optional general turnaround helpers, with the restriction that one electrical specialist may supervise no more than (for example) four helpers. Specialists typically have a higher cost per hour than general turnaround staff and so it is useful to have them supervise helpers, within limits. Similar computations and constraints are constructed for each of the tasks and labor specialties. The numbers of each of the specialist types and general turnaround staff are decision variables in an optimization whose objective is to achieve a minimal cost staffing level which achieves the desired flight rate, that is, ensures that enough manhours are provided of the differing types to satisfy the needs of the various subsystems and processes. If the turnaround facility is open 250 days per year at two

shifts per day, then a staffing level of one electrical specialist per shift will provide 4000 hours ($=250*16$) per year with which to satisfy the needs of the various tasks which require electrical specialists.

The formulation that arises from the above considerations is a mixed integer program. This formulation is modeled in a third worksheet of Hyperport.xls and is solved using Excel Solver. The resulting initial recommendations for numbers of line personnel can optionally be copied back to the user input sheet and then simulated. The actual discrete event simulation model by Hyperport is vastly more complex than the initial line personnel recommender and we do not expect in general that the target flight rate specified to the manpower recommender will be achieved in the Arena simulation. For example, a target flight rate of 12 flights/year specified to the initial manpower recommender might result in manpower allocations which, when simulated in Arena, actually result in a flight rate of 9.2 flights/year. In this case, the user might then subsequently specify a target flight rate of $(12/9.2)*12 = 15.7$ flights per year to the initial line personnel recommender and then run the Arena simulation with the resulting manpower allocations. With this method, the user can achieve in a handful of iterations suitable line personnel allocations to use as the basis for further manual manipulation.

ONGOING WORK INCLUDING DESCARTES-ORIGINS

Since the initial completion of version 1.0 in January 2009, Hyperport has been the centerpiece of SEI's operations modeling capabilities. Version 1.2 has already been utilized for two different Air Force-sponsored vehicle concepts. In both cases, it was a part of a 'Level 1' operations analysis, comprised of Hyperport results for a stated desired flight rate, depot cost estimates, and a package of data including some custom metrics highlighting primary operations cost drivers. This was integrated with reliability work and non-recurring cost estimates to give full life-cycle cost analysis to the customers sponsoring those efforts. Additionally, by time of publishing, SEI will have completed a 'Level 2' analysis of one of these vehicles, and be working on a Level 1 analysis of a NASA concept. The Level 2 work will incorporate a wider exploration of trades using the Hyperport outputs as the key metrics, along with a higher level of fidelity in many of the supplemental estimates. Additionally, larger numbers of replications at Level 2 will allow more output of more complete

distributions and/or percentile data, beyond the standard mean and confidence interval.

It is the intention of SEI that we will continue to use Hyperport in its current form for this sort of work, and with it be able to provide meaningful estimates of recurring costs and turnaround times for future reusable launch vehicle concepts. It is also possible, given the modular nature of the Arena model, that if higher fidelity was ever required for a particular subsystem, facility, or other cohesive set of processes, that the SEI team could build upon the existing submodel and incorporate those extensions into the larger model. Moreover, if and when we become aware of additional data sources, they will continue to be integrated into the DescartesData.xls spreadsheets. While the specific steps taken at any given time may be dependent on a particular customer's needs, they will collectively lead to ongoing improvement in the accuracy of Hyperport results. Finally, as stated in the introduction, it is also a long-term objective to make Hyperport (by way of its .xls files) compatible with SEI's 'program metrics' integrated ModelCenter model I-RaCM.

Descartes-Origins

While it is likely that eventually new versions of Descartes-Hyperport will be released, the NRA-funded research has progressed into Phase II. Hyperport is actively being used for analysis tasks, but the Descartes team has now shifted its focus to solving the second half of the problem described in the introduction. That is, while we can now demonstrate knowledge of which vehicle concepts are more or less expensive to operate, we would like to be able to use comparable methods to determine net effects on total life-cycle cost. The second Descartes model, Descartes-Origins, will be the tool needed to answer these questions.

SEI currently utilizes several industry-standard cost estimating tools including the NASA-Air Force Cost Model (NAFCOM), SEER-H, and TransCost. While these have all been considered generally successful, it is the opinion of the authors that similar weaknesses exist in these models to those in most current operations models. Namely, they are overly-reliant on building analogies to historical programs. Additionally, in this case, they cannot readily produce cost estimates that are integrated with estimates of time. A DES model is extremely well-suited to pair these types of metrics, as demonstrated by Hyperport.

Origins will be built to model the various phases of design, development, and production of reusable launch vehicles. Research so far has been focused on generalizing the design and development sequences used by NASA and the US Air Force (or, more generally, the Department of Defense). As shown in Fig. 8, the two agencies have varying checkpoints and design reviews, but that the flow is easily generalized into distinct phases. Each of these phases will likely become a submodel in Origins.doe, with entities representing a concept design (later becoming the concept in production, testing, etc.) moving through them in sequence. Depending on the design phase, there will be resources representing internal design teams and/or contractors, and later on things such as test and production facilities. The submodel or submodels related to vehicle production will be further broken into vehicle systems. Users will have the option of assigning 'black box' time and/or cost estimates to particular parts that may come from a reliable supplier, and simulating others that are newer technologies or that simply have more unknowns.

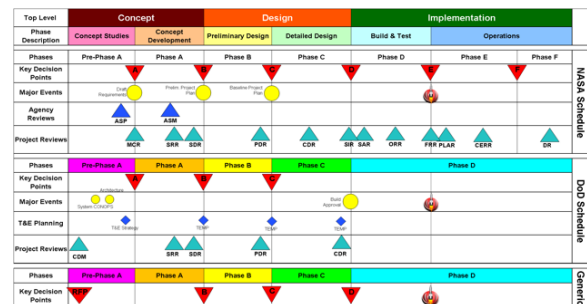


Figure 8. NASA and Department of Defense vehicle design processes

Research has been performed on a number of past programs to gain insights into how the design and production flow have evolved, and how closely they are adhered to depending on a variety of circumstances. The cases studied include successful programs with numerous launches, X-vehicles with very limited production, some programs that were cancelled entirely, and recent private efforts such as the Space-X Falcon 9 that are receiving some government funding. This wide variety will allow SEI to build a model that can be responsive to high-level decisions concerning funding levels, intended vehicle applications, levels of contractor involvement, sponsoring agencies, and whether or not the vehicles are to be manned. In response, the DES capabilities will be able to output probabilities of success, along with estimated costs and times for both successful and unsuccessful programs. The

nature of the work being modeled is very different from in Hyperport, and so the probability distributions are likely to be different as well. Research to date has indicated a very asymmetric trend wherein projects very rarely reach deadlines ahead of schedule, or complete a project under budget, but delays are common, as are cost overruns. When possible, we will study not only how long programs took and how much they cost, but what their initial projections were, and why those were exceeded.

Once completed in late 2010, Descartes-Origins will be able to give estimates of DDT&E and TFU costs in a similar form to those currently available from other tools, but they will be coupled with time estimates and chances of success, giving a more complete picture than is currently available of the lifetime potential of a future program.

CONCLUSION

The Descartes project's goal is to be able to consistently estimate reusable launch vehicle project costs and timelines, using a single coherent modeling philosophy. The ability of discrete event simulation to model complex systems with numerous process flows and interaction of resources makes it an ideal choice for this type of work. Descartes-Hyperport has become a working tool that is being employed on an ongoing basis. Hyperport has performed well as a proof-of-concept, producing high-level estimates in line with expert opinions while also giving insights into lower-level cost drivers and sensitivities that may not have previously been available. Descartes-Origins will be able to take advantage of lessons learned and modeling techniques perfected while developing Hyperport to become an equally helpful tool contributing to life-cycle cost analysis.

As the Space Shuttle fleet is retired and a variety of private companies and government programs work towards the next generation of reusable launch vehicles, decisions are likely to be made based not only on vehicle performance, but also on cost, reliability, and long-term maintainability projections. The tools developed at SEI during the Descartes program will be able to provide reliable, detailed estimates, giving programs the data they need to make those decisions. Hyperport is already supporting this objective, with Origins following soon.

ACKNOWLEDGMENTS

Sponsorship and financial support for major portions of this project are provided by NASA's Aeronautics Research Mission Directorate under 2007 NASA Research Announcement contract NNL08AA30C. The authors would specifically like to thank the Contract Officer's Technical Representative Mr. David Rudy, the Technical Monitor for the project Ms. Lakisha Crosby, the hypersonic technical point of contact Mr. Jeffrey Robinson, Mr. John Martin, and all of NASA Langley Research Center.

The authors gratefully acknowledge the contributions of colleagues at SpaceWorks Engineering, Inc. who contributed to this paper. Dr. Ron Menich (Space Systems Analyst at SEI) assisted with the latest version of Hyperport, and interns Mr. Brendan Dessanti and Mr. Luke Walker supported research for the Origins model. Mr. William J.D. Escher (SEI Affiliate) and Mr. Carl Ehrlich (SEI Affiliate) provided sage advice as well as references relevant to operational aspects of past aerospace programs.

REFERENCES

1. NASA SP-407, Space Shuttle, ca. 1976; p. 4.
2. Kennedy Space Center, "Space Transportation Ground Processing Operations Modeling and Analysis: A Review of Tools and Techniques," Edgar Zapata, Systems Engineering Office, NASA KSC, June 2003, p. 6.
3. DePasquale, D., Kelly, M., Charania, A., and Olds, J., "Activity-Based Simulation of Future Launch Vehicle Ground Operations," AIAA-2008 7660 Space 2008, San Diego, California, September 9 - September 11, 2008.
4. DePasquale, D., Charania, A., "I-RaCM: A Fully Integrated Risk and Life Cycle Cost Model," AIAA-2008 7703, Space 2008, San Diego, California, September 9 - September 11, 2008.