Proceedings of the 2002 Winter Simulation Conference
E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds.

GENERIC SIMULATION MODELS OF REUSABLE LAUNCH VEHICLES

Martin J. Steele

YA-D6
Kennedy Space Center
KSC, FL 32899, U.S.A.

Mansooreh Mollaghasemi
Ghaith Rabadi

Productivity Apex, Inc.
3403 Technological Avenue, Suite 7
Orlando, FL 32817, U.S.A.

Grant Cates

PH-M3
Kennedy Space Center
KSC, FL 32899, U.S.A.

ABSTRACT

Analyzing systems by means of simulation is necessarily a time consuming process. This becomes even more pronounced when models of multiple systems must be compared. In general, and even more so in today's fast-paced environment, competitive pressure does not allow for waiting on the results of a lengthy analysis. That competitive pressure also makes it more imperative that the processing performance of systems be seriously considered in the system design. Having a generic model allows one model to be applied to multiple systems in a given domain and provides a feedback mechanism to systems designers as to the operational impact of design decisions.

1 INTRODUCTION

The typical scenario for a simulation study entails developing a specific model of an existing system for the purpose of analysis. This begins with the capturing of knowledge by the simulation analyst from the system expert, including information on system structure and data, and continues to the modeling of that information and data at some level of abstraction, the running of model scenarios, and the subsequent analysis of the resulting model output data. Typically, this is accomplished of a system that is in existence, or nearly so, that has the requisite details known by which to construct and run a simulation.

Conducting a simulation study becomes more complicated under two conditions: first, when details of the system are as yet unknown because it is early in its design phase, and second, when two or more competing systems are to be compared. It is important to emphasize that this second condition is different from the comparison of competing configurations of the same system. In comparing competing configurations, the same model may be used; however, in comparing competing systems, different models are typically required. This necessitates the development of separate simulation models of the competing systems with all the time and effort involved in each. In addition, it is important in these circumstances to ensure the simulation models are constructed at equivalent levels of detail and abstraction to avoid introducing dissimilar perturbations to one model over another.

The problem here is that it can take months to study a system by the use of simulation modeling. Developing multiple system-specific models becomes problematic due to time and budget constraints. Some of the steps involved in simulating multiple systems will be shared because of the common objectives; however, many of the most time consuming steps would be unique to each system modeled. Therefore, the time for modeling and studying each individual system becomes additive to the overall effort.

In a majority of cases, design projects are not willing to wait for lengthy analyses. Especially in today's competitive global marketplace, months of analysis induces too large a lag in production capability. Additionally, technical and market growth and change can make a system obsolete prior to operation. Finally, if system performance analysis is too lengthy to be useful, the opportunity to influence the design of a system with respect to the operational impacts a particular design decision is eliminated. On, the other hand, if it is possible to show the operational implications of design alternatives through simulation in a timely manner, it may be possible to influence the design towards a more operational system. The benefit is that

these operational improvements/benefits will be experienced iteratively throughout the operational life of a system. From the life-cycle perspective, operational costs far exceed those of design. So, preventing, eliminating, or reducing the recurring costs of a system will be realized in each year of operation and may far out-weigh the related design costs.

The question is then, how to develop a valid and credible model

- Of a non-existent system or a system in design,
- In a timely manner, and
- Compare it with similar models of competing systems

To this end, it is helpful to start with a baseline or benchmark, or at least a similar system.

One consideration is to develop a generic simulation environment of the domain at hand. Because this approach is more general, additional care must be taken in its development to ensure its applicability to all instances in the chosen domain, while also capturing enough specific system detail so as to be credible. This has two benefits. First, though a more generally applicable model is more difficult to construct and validate, the result may be used (amortized) across more than a single implementation. Second, the time required in the simulation study should be reduced because the model need only be populated with data from various systems and not constructed from scratch.

2 LITERATURE REVIEW

The research accomplished in the area of generic simulation models falls into two primary areas: that of developing models applicable to more than one system and of simulation models that are uniquely composed from a library of previously developed modules. Both approaches have their associated benefits and detriments, but those are strictly relative to the purpose to which each method is applied.

One of the risks in developing systems of any kind, and simulations of systems are no exception, is requirements creep, i.e., the steady growth of requirements as the system is being developed. Projects subject to this single ailment almost always exceed their projected budget and schedule, and many do not continue to fruition. This is experienced in simulation systems as developing beyond the initial required level of abstraction. In other words, lower levels of detail are added beyond that required to, for example, experiment with the overall throughput of the system under study. Two ways of avoiding this pitfall and producing the results of a simulation study in a timely manner is by strictly enforcing only the requisite level of detail or constricting the scope of study to a manageable size. To this end, Brown and Powers (2000) limit their F-16 Air Force Wing simulation model to those components and processes required for operations and maintenance. To further concentrate their efforts, the focus was specific to the maintenance that had a critical impact on operations. This "simulation in a box," as they call it, is generic in its applicability to something on the order of an Air Force wing operation. Other important issues in any type of simulation modeling that are discussed include the following.

- Focusing on important factors – this helps to manage the complexity of the model and helps to keeps costs in-line.
- Simplifying input – the input of knowledge and data to the generic simulation model should be easy for the user and widely available
- User-friendly output – graphs and charts should be what is needed and useful to the user

Mackulak, Lawrence, and Colvin (1998) also make the case for generally applicable models that can be configured to more specific cases in the design analysis of Automated Material Handling Systems (AMHS). Such design environments require the quick turnaround of analyses to support the production of their AMHS's at rates up to one to two per week. The more traditional development of a simulation model specific to a single system simply does not meet the need of this environment. The IntelliSim project (Mackulak and Cochran 1990) indicated that 45% of a simulation project's effort is in model building, formulation, and translation. Therefore, implementing a generic model that is applicable to many systems can save a significant portion of time in a simulation study. In addition, such models, once developed, are already debugged and can be optimized for quicker run times.

The area of composing simulation models from a library of previously developed modules is generic at the module level but is more specific to the system under study. This method too can save time and money in a simulation study, and result in a system that is less abstract than a wholly generic simulation. This has the added benefit of being easier to validate because the resulting model can be more recognizable than the system-level generic model. Many considerations between the two methods are similar, but some are applied at the system level, while others are applied at the module level.

The Winter Simulation Conference of 2000 had a session on composable and reconfigurable simulations. All of the papers (Diaz-Calderon, A., Paredis, C. J. J., and Khosla, P. K. 2000; Kasputis, S. and Ng, H. C. 2000; Davis, P. C., Fishwick, P. A., Overstreet, C. M., and Pegden, C. D. 2000; Son, Y. J., Jones, A. T., and Wysk, R. A. 2000) addressed the importance of less expensive and quicker simulation analysis that is possible with this type of simulation development and that also promotes model or more correctly module reuse. The requirement for this approach, however, is that such a library of modules must exist. The purpose of this study is directed towards a system level generic model.

3 METHODOLOGY

Implementation of the steps in a simulation study can take on many forms and are discussed in the textbooks of discrete event simulation. However, several key steps become important when attempting to construct a simulation model that is generic to all the systems in a broader domain.

The first step is to select the *domain* of interest. The considerations here are similar to those of addressing the problem to be analyzed but with a broader perspective. The problem and domain must be identified to adequately address the objectives of the study. It is important to include only the requisite details of the systems in the domain. It is easy when working with system experts to include lower levels of detail than necessary for the objectives of the simulation project. Keeping only the appropriate level of detail is another aspect of abstraction in modeling. On the other hand, a domain that is too broad can lead to exceedingly large models that become difficult to understand and maintain.

The second step is to draw a *conceptual level diagram* of a generic system in the domain. These diagrams show the processes of the systems and their interrelationships as entities flow through the system (an example is shown in the following section). Developing a generic model from scratch can be overwhelming (Brown and Powers 2000) and so it can be beneficial to start with a representative system that is subsequently broadened to the domain level. Once a system specific conceptual diagram is constructed, careful consideration of the processes and entities is required to make the shift to the generic level. This is the point where model abstraction increases and face-validation with system and domain experts becomes critical. Implementing abstraction in a model is the realm of the simulation analyst, but is typically contrary to the thinking of system experts. In order to keep model size and focus under control, communication of system abstraction is crucial to maintaining model validity from the perspective of the system expert and is always an iterative process. The trick here is to include enough detail to be useful, but have enough abstraction to be generic.

Once the conceptual diagram is completed, it should be a fairly straight forward task to *identify the constructs* that make up the system and relate them to the constructs of a simulation model. The constructs of a system process are, generically, things that flow through the system, actions that occur to those things, and assets that are required to perform the actions. These system or domain constructs must be mapped to the constructs of a simulation model (Table 1) for the translation of the system/domain into a computer simulation model.

Translating the conceptual model into a computer simulation model is at this point typical to other simulation studies and so is mentioned only for completeness. Processes, entities, and resources are defined and related in the simulation environment of choice with generic variable names, so that specific system data may be loaded for running of the model with information from a system within the domain.

Table 1: Simulation Model Constructs

Entity • Arrival time model • Attributes • Sequence/Routing – Per entity type – Time
Process (Server) • Process time model • Queue size & protocol • Resource requirements
Resource • Types • States • Daily Schedules • Maintenance Schedules • Reliability (MTBF) • Quantity
General Simulation/System Characteristic • Terminating • Steady State

Another key aspect of developing generic models is in the method of *eliciting* the requisite *information* from the now broader scope of many different system experts within the selected domain. More traditional simulation model development obtains this information on a very limited scale, specific to one system. What is interesting at this point is to elicit this same information, but of various systems that may use very different terminology. One of the tasks of any product or service is to get the product used. To this end, the user interface must allow the system experts to enter information into the generic simulation using terminology with which they are familiar. Subsequently, this same system specific terminology should also be used in displaying the output results from running the model. The user interface must, therefore, provide a structure by which information is solicited, independent of the underlying generic model. Entry of this information should allow specification of specific system components in the terminology native to that system. Because this generic environment will be used on many different systems, a simple and intuitive user interface is imperative.

This methodology resulted from the development of a generic simulation model of reusable launch vehicles (RLVs) for the National Aeronautics and Space Administration (NASA). This example is discussed in the following section.

4 AN EXAMPLE GENERIC MODEL

NASA's Space Launch Initiative (SLI) program is currently studying various architectures for the next generation reusable launch vehicle (RLV). Several competing companies are developing designs that address the aggressive requirements of that program in the area of cost, reliability (safety), and availability with the goal of having magnitudes improvements over the first generation RLV (i.e., the Space Shuttle). When these proposals are submitted, it is necessary to compare the competing designs by as similar means as possible, and in a timely manner.

The concept of availability is directly related to a specified launch rate, and is also a direct factor in the economical case for any launch vehicle. Because of the great expense of acquisition and operation of space launch vehicles, one way to justify the expenditure is with the ability to amortize those costs across a greater number of launches. The maximum launch rate achieved by the Space Shuttle was approximately 10 flights per year, but this was not for a sustained period. So, one way to improve the economical argument for the next generation RLV is to increase the number flights that vehicle is able to make in a given year. A way, one may argue, the best way, to analyze the operations performance of a RLV is by discrete event simulation (DES).

For any system, the issue of performance is multifaceted and may be viewed on three general levels. The first is does it work at all. This might be considered the *physics level* of performance in that the system is able to work to any degree (e.g., the system does something, the system works correctly). Secondly, at the *engineering level*, the concern is for how well the system works. Systems always have some level of performance specification to which they are designed (e.g., a communication system must transmit data at so many bits per second, a structural system must withstand so much force, a rocket must produce so much thrust in order to lift so much weight). The world of design engineering is saturated with such thinking, and for some systems this is to the exclusion of operations performance.

This is also typically dependent on the maturity of a given technology and the market demand. The Space Shuttle is a first generation RLV and so the primary focus of the program was to simply make it work to some level of engineering specification. To this end, it is successful with over 100 flights to its credit. However, the operations performance, that is, the desired flight rate and resource requirements for turnaround processing, is not what was desired. Program requirements for the second generation RLV place a much greater focus on operational performance.

The issues precipitating the need for a *generic* RLV simulation model were to analyze the operations performance of several architectures in a timely manner, and to provide feedback to the design community as to the operational ramifications of design decisions. To this end, the Generic Simulation Environment for Modeling Future Launch Operations (GEMFLO) was developed.

Following the methodology of the previous section, the first task is to determine the *domain* of interest. When starting this project, the goal was to develop a generic model applicable to any *reusable* launch vehicle or launch vehicle with both reusable and expendable components. Upon completion of the first phase of the project, it is possible that GEMFLO can simulate any launch vehicle turnaround process (reusable or expendable), but this has not yet been verified.

In developing the generic *conceptual flow diagram*, the diagram used in developing the simulation model of Shuttle ground processing (Cates, Steele, Mollaghasemi, Rabadi 2002) was used as a starting point. This proved a very useful exercise, required several iterations, and required some additional abstraction than would be found in a model of a single system. What resulted from this expansion from specific system level to domain level is an understanding of the processes that are generic to any launch vehicle flight hardware element (FHE). Table 2 lists the processes in order, arbitrarily starting with ascent. The conceptual flow diagram is shown in Figure 1.

Table 2: Generic Flight Hardware Element Processes

1.	Ascent
2.	Staging
3.	On Orbit
4.	Landing Site Decision
5.	Descent
6.	Landing
7.	Post-Flight Safing
8.	Transport
9.	Post-Flight Processing
10.	Depot Maintenance Decision
11.	Normal Processing
12.	Integration
13.	Launch

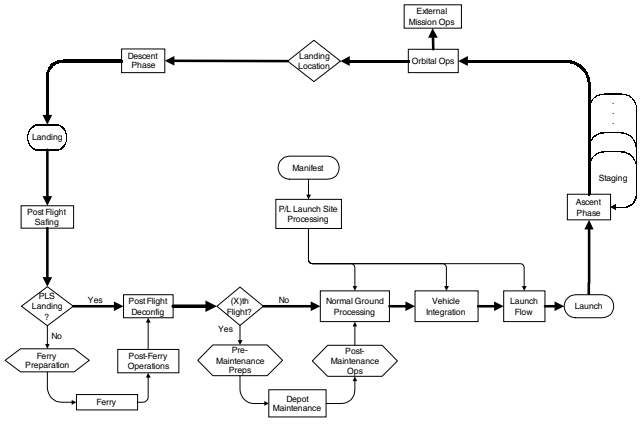

Figure 1: Generic RLV FHE Conceptual Flow Diagram

It is important to point out one manner of abstraction in this diagram for reusable or partially reusable launch vehicles. Taking the Space Shuttle as an example, the Solid Rocket Boosters (SRBs) and External Tank (ET) do not reach orbit (i.e., reach the orbital operations process). Yet, after staging of the individual FHEs, they all show the similar flow of going "on orbit." This is a level of abstraction required for this model to be generic, but is compensated by the fact that zero process times are inserted for those generic processes that in reality do not occur. This keeps the overall flow generic, but does not affect the statistics kept while running the model. As another example, some future RLV may have a single stage to orbit architecture. In the conceptual flow, *integration* of FHEs and *staging* during ascent would *not* occur in reality, but in the running model, they would occur with zero delay or process time.

With the conceptual flow diagram in hand, the RLV processing constructs are enumerated. These include FHEs, ground processes, and ground resources, examples of which are shown in Table 3. This information is what is *translated* into a generic simulation.

Table 3: Generic RLV Simulation Construct Examples

FHEs	Ground Resources for
Crew Vehicles	Processing
Payload Vehicles	Integration
Boosters	Launch
Tanks	Landing
Payloads	Retrieval
	Safing

In conjunction with the development of the generic simulation model, a mechanism for eliciting the requisite information from individual system experts to input to the simulation must also be developed. As previously discussed, this is a crucial part of the generic system development in that the intuitive appeal of the product will go a long way to getting it used, thus, amortizing its longer development time across a greater number of customers. For this particular product a Visual Basic user interface was developed. It starts out with generic RLV terminology in requesting the number of FHE included in a single RLV, but allows the naming of those elements and other related resources to user specific instances. In this way the results of a simulation run will show, for example, utilization of an Orbiter Processing Facility for processing the Shuttle instead of just a generic processing resource. The underlying model is generic, but it is populated with system specific information so that the results are reported as if a system specific model were developed.

One measure of validation of this system was to take the data from a previous simulation model of Space Shuttle processing and use it as input to this generic RLV model. The results of both the generic and specific models were the same. Though this is encouraging, it is as yet only one data point. Continuing use and development of this generic system will refine these results.

One difference between the specific Shuttle simulation and the generic RLV simulation is that the FHEs were treated differently in each model. The previously developed Shuttle model treated the FHEs as entities that were processed by the system. For example, the orbiter enters the system and is processed in the orbiter processing facility. In the generic model, FHEs are treated as resources that are requested by certain processes to service the *mission* entities that are created by a defined input process. This second approach is a more accurate and more useful rendition because it allows the collection of utilization statistics on the FHEs and the experimentation with the quantities of FHEs in the fleet. This will be a great assistance in determining the correct complement of flight and ground resources to meet a given mission manifest.

5 DISCUSSION

The investment in generic models of many systems within a given domain appears worthwhile, with supporting cases from the F-16 Wing and AMHS venues. The case for the generic RLV simulation model will be made with application to the forthcoming next generation RLV designs.

Comparing the generic development process with that of specifically focused simulation models points out the values of each approach. This comparison of the different aspects is given in Table 4.

It is especially beneficial to organize and communicate the pertinent aspects of operations performance to the design community. From the design process, bad decisions are paid for and good decisions are reaped in the much longer operations life-cycle of any product. Generic modeling is one method that can assist in communicating the operations ramifications of design decisions.

6 CONCLUSIONS

The longer development time of generic models is outweighed by the benefit of amortizing the use of those models across many systems in the chosen domain and the shortened experimentation phase once the mode is developed. Generic models also offer the opportunity to organize operationally pertinent information on systems in a domain so they may be compared on a more level playing field than is easily possible with differing models. The addition of domain specific details could enhance the fidelity of generic models if deemed necessary. Specific models do provide an easier path to higher fidelity analyses and more representative animation, which can be crucial to obtaining face validity.

Table 4: Specific & Generic Aspect Comparison

Aspect	**Specific Sim.**	**Generic Sim.**
Development	Focused on a single system	Attempts for general application make development more complicated; requires more time
Domain	Narrow – Single System	Broader
Abstraction	More system specific – can get as detailed as desired	More abstract relative to any one system
Animation	As much as desired by developer & customer	Only minimally present unless manually added for each specific system addressed
Validation	Standard simulation techniques	More difficult to validate due to higher degree of abstraction
Use	Can be easier due to focusing on a single system. May be more difficult if ease of user input not considered.	Most likely easier due to the need for structured knowledge capture in the generic venue.
Invested Time	Development time is shorter but must be repeated for each system modeled	Development time is longer but has broader application (Mackulak, et al. 1998). Once developed, time required of the simulation study is shorter than that required in developing and analyzing the single system simulation

REFERENCES

Brown, N. and Powers, S. (2000). Simulation in a Box (A Generic Reusable Maintenance Model). In *Proceedings of the 2000 Winter Simulation Conference*, ed. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 1050 - 1056.

Cates, G. R., Steele M. J., Mollaghasemi, M., and Rabadi, G. (2002). Modeling the Space Shuttle. In *Proceedings of the 2002 Winter Simulation Conference*, ed. *E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes*.

Davis, P. C., Fishwick, P. A., Overstreet, C. M., and Pegden, C. D. (2000). Model Composability as a Research Investment: Responses to the Featured Paper. In *Proceedings of the 2000 Winter Simulation Conference*, ed. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 1585 – 1591.

Diaz-Calderon, A., Paredis, C. J. J., and Khosla, P. K. (2000). Organization and Selection of Reconfigurable Models. In *Proceedings of the 2000 Winter Simulation Conference*, ed. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 386 – 392.

Kasputis, S. and Ng, H. C. (2000). Composable Simulations. In *Proceedings of the 2000 Winter Simulation Conference*, ed. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 1577 – 1584.

Mackulak, G. T. and Cochran, J. K. (1990). The Generic-Specific Modeling Approach: An Application of Artificial Intelligence to Simulation. In *1990 IIE Integrated Systems Conference & Society for Integrated Manufacturing Conference Proceedings*, 82 – 87. San Antonio, TX: IIE.

Mackulak, G. T., Lawrence, F. P. , and Colvin, T. (1998). Effective Simulation Model Reuse: A Case Study for AMHS Modeling. In *Proceedings of the 1998 Winter Simulation Conference*, ed. D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan, 979 – 984.

Son, Y. J., Jones, A. T., and Wysk, R. A. (2000). Automatic Generation of Simulation Models from Neutral Libraries: An Example. In *Proceedings of the 2000 Winter Simulation Conference*, ed. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 1558 – 1567.

AUTHOR BIOGRAPHIES

MARTIN J. STEELE (Ph.D.) is an engineer with NASA at the Kennedy Space Center (KSC) with a wide range of experience, from shuttle and payload operations to ground systems and facilities development. He is currently leading several efforts at KSC to employ simulation modeling in the operations analysis of existing and future launch vehicles. His research interests include simulation modeling and analysis of complex systems, simulation input modeling, generic system simulations, and neural networks. His email address is <martin.steele-1@ksc.nasa.gov>.

MANSOOREH MOLLAGHASEMI (Ph.D.) is the president and CEO of Productivity Apex, Inc. and an associate professor in the Industrial Engineering and Management Systems Department at UCF. Her research and teaching interests include simulation modeling and analysis of complex systems, statistical aspects of simulation and simulation optimization, operations research, probability and statistics, neural networks, and multiple criteria decision making. Her email address is <mmollagha@productivityapex.com>.

GHAITH RABADI (Ph.D.) is an assistant professor at Engineering Management Department at Old Dominion University. He was a visiting assistant professor at the Industrial Engineering Department at the University of Central Florida, and a vice president for R&D at Productivity Apex, Inc. where he managed simulation and risk analysis research projects funded by NASA. His main research interests are Operations Research, Scheduling, Simulation, and Machine Learning. His email address is <grabadi @productivityapex.com>.

GRANT R. CATES has 20 years of experience working on the space shuttle in various capacities including construction and activation of the launch complex, payload integration and processing, and space shuttle vehicle ground operations. His email address is <grant.cates-1@ksc.nasa.gov>.