*Proceedings of the 2010 Winter Simulation Conference*
*B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, and E. Yücesan, eds.*

# MODEL FLEXIBILITY: DEVELOPMENT OF A GENERIC DATA-DRIVEN SIMULATION

Nick A. Brown

KLSS
2091 Exchange Ct.
Fairborn, OH 45324, USA

## ABSTRACT

The idea of simulation model "re-use" is a novel term that in theory will allow for quick turn-around times where budgetary constraints can hold back the development of a new model. The intention of this paper is not to examine a specific example of how a simulation was developed and utilized for "re-use", but rather explain the process of developing a computer simulation flexible enough that will allow for "re-use". The overall outcome of this type of development is a data-driven simulation model that is flexible enough to expand to many similar systems without significantly altering the code of the simulation. As a result of this data-driven simulation, companies/organizations are able to reap the benefits of reducing future development time, utilizing the model for other similar systems, achieve quick turn-around, and the ability to perform large scale sensitivity analysis.

## 1 INTRODUCTION

A model is considered generic when the model has enough flexibility to allow for modeling of similar systems by altering the input data used to execute the model. This kind of simulation model will be generic in logic only and will be defined through input data. Combining the terms flexibility and generic contribute to model "re-use". The "re-use" model is defined or made specific when the data that is utilized to execute the model is populated.

There are differing views of developing a model for "re-use", which is observed in the literature review. There are many issues in developing a model for "re-use" as shown in papers written by Overstreet, Nance, and Balci (2002) and Paul and Taylor (2002). However, there are some examples that contradict that "re-use" is not feasible. In papers by Brown and Powers (2000) and Mackulak, Lawrence, and Colvin (1998) specific examples of a model developed for successful "re-use" are presented. What is evident from these papers is that there is only a small mentioning of how to develop one of these flexible models. There is some mentioning of items such as developing a focus/plan, develop logic into submodels, common libraries of code, and planning for expandability. All of these papers only mention these topics, but no in depth discussion of how to design a model for flexibility. This paper differs from the papers reviewed, in that this paper describe the phases in developing a generic flexible simulation model for "re-use" as opposed to describing development for a specific application.

The phases presented in this paper are similar to those adhered in developing any simulation. The foundation of this paper is to present these defined development phases expanded for development of a flexible generic model for "re-use". The phases are presented in chronological order, where information and decisions made in one phase are used for subsequent phases. It is important to note that all areas of each phase may not be utilized and some steps could be more situational. Though each step in a phase may not be performed, each phase must be performed. In developing a model for "re-use" the developers

should first proceed through the planning phase, then the software selection, organize the data structure, and finally create the simulation code (Figure 1).

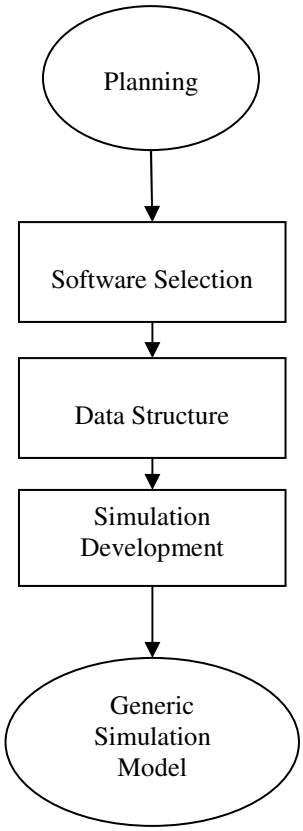


Figure 1: Generic simulation development steps

The intention of this paper is to discuss methods for developing generic data-driven simulation models. Through the presentation of phases/steps/tips in this paper modelers will have a foundation or set of rules for creating more dynamic models. These dynamic models can then be used in other instances. While developing a model an important question to ask is, "How can I make this flexible enough to achieve the goal of creating a generic data-driven simulation model?" This question will appear many times throughout the paper to emphasis the importance of reviewing the current task and then how it can be made flexible.

## 2 LITERATURE REVIEW

The topic of model "re-use" has been receiving more attention and has become a central objective of High Level Architecture (HLA) with-in the defense sector (Overstreet, Nance, and Balci 2002). A lot of attention on how to create models for "re-use" has been initiated in the defense sector because any simulation developed under Small Business Innovative Research (SBIR) are required to be applicable commercially (Brown and Powers 2000). Additionally, successful development of these models trim down cost expenditures due in part to reduced development time. The most important development of a model for "re-use" is the capability to eliminate the overabundance of models that are developed for very similar systems (Brown and Powers 2000). But, while there are organizations pushing for model "re-use" and examples of companies that have developed these types of models. There are some researchers and organizations that dispute the model "re-use" idea. This section will review papers that both support and dispute the topic. The papers vary from one paper reviewed stating that re-use is not useful; while other papers

present specific instances of how companies have successfully developed models for "re-use". One successful model development is focused in the commercial world, while the other focuses on the department of defense.

In a paper written by Overstreet, Nance, and Balci (2002) the authors describe some issues that are inherent in developing a simulation for "re-use". The main issues of "re-use" in simulation are that simulation models are built around particular objectives and constraints (Overstreet, Nance, and Balci 2002). Simulations are designed to model just enough to get accurate results and to answer an exact question (Overstreet, Nance, and Balci 2002). On the other hand developing sub-models, which is code developed as models within the model, can be used interchangeably in other models to promote "re-use" (Overstreet, Nance, and Balci 2002). Paul and Taylor (2002), echo most of the issues by Overstreet et. al (2002), but claim that it might cost companies more to re-use a model as compared to building an entire new model from scratch. The one common theme of Overstreet et. al (2002) and Paul and Taylor (2002) is the idea of shared libraries or common code between models, sub-models.

While the Overstreet, Nance, and Balci (2002) and Paul and Taylor (2002) papers review the issues and the assumption that "…model reuse is no use" (2002); papers by Brown and Powers (2000) and Mackulak et al. (1998) make a case for model "re-use". In the paper, Simulation in a Box (A generic reusable maintenance model), Brown and Powers (2000) begin by developing a maintenance model called SIMFORCE. In the development process they attempt to define what a generic model is. Brown and Powers (2000) have defined a generic model as a model that can be altered through data and be used for other applications with common reports. Their process in the development of SIMFORCE was through specific model definition (maintenance F-16 unit), but also considered ways to expand the functionality of the model (Brown and Powers 2000). The steps that were used in model definition were to define the focus of the model, where to add flexibility to the input data of the model, what fidelity to scope the model, and final consideration of output. The data was flexible enough to model for large set of data, where the model in the current state only included a small subset. This allows for flexibility to include more/less data elements in certain areas.

In a paper written by Mackulak et al. (1998) the authors are using a model developed for automated material handling systems (AMHS) to back their claim that a "…generic model could be reused, thereby reducing model building time…". In this effort the design of the simulation of the AMHS must be highly flexible due to quick turnaround times. These quick turnarounds are necessary to show potential customers the benefits of the system. An individually designed model for clients is not a choice. They also have there own definition of generic, which is very similar to Brown and Powers (2000). The definition they have included is that a model is generic and only becomes specific when populated with data (Mackulak et al. 1998).

These papers are the base for which this paper will argue for the development of a flexible generic simulation model for "re-use". The papers make arguments to both sides and while the issues are genuine the general statements that model "re-use" is not feasible is not completely proven. There is no method in any of the papers that give a definitive method for developing a model for "re-use". The proceeding sections will detail the phases that have been used to create flexible generic models. Where through altered data a simulation can be changed to model similar systems.

## 3 PLANNING

Planning is one of the most important phases in the development of a simulation, but is often the most overlooked and forgotten. As it so often happens this phase of development is skipped for many reasons, but the most viable reason has to do with getting everyone involved in the project together. Without this critical step there almost always exists a disconnect between one or more parties. The developer is already at a disadvantage in the development process when this step is skipped. As opposed to developing for a purpose and defined direction the developer is now reliant on very little to no information and must begin without clear direction. Without meeting with everyone involved the model must be created with a increasingly larger scope, which adds time to development. The developer is still able to develop a simu-

lation without the clear direction developed in this meeting, but the other steps in this phase become much more critical. The following subsections will describe the steps to take in the planning phase. The first step, meeting with all involved parties, is an ongoing step throughout the planning phase; while the other steps are mostly accomplished in conjunction with all involved parties.

**Step One: Involve all Parties**

During this step it is advisable to assemble a team of direct members of the development team. This includes program management, end-users, simulation developers, and data collectors. The following steps of the planning process should be performed with all involved parties in the project.

**Step Two: Model Objective**

The next step is to develop an objective of the model. What question the model is going to answer. It seems pointless to develop a simulation without the idea of what question the simulation is trying to answer, but this happens more often then not. There needs to be a clear definition as to what needs to be answered.

**Step Three: Scope the Model**

Answering the question in step two, what is the objective of the model, will generally move into how to model the simulation? As Overstreet, Nance, and Balci (2002) stated the goal of a simulation is to model just enough to get the results by minimizing simulation development and run length. When in doubt of how to scope the model it is important to ask the question again, "How can I make this flexible enough to achieve the goal of creating a generic data-driven simulation model?" If the question still does not answer how to scope the model then the objective of the model could be vague (go back to step two). Consider revising the objective to make it more concrete. Next, consider constraining factors for developing the scope of the model. Constraining factors are a good starting point when you are unsure of the scope. These factors are those variables that will be altered in performing sensitivity analysis. These are also very important in defining what data will be required. The data elements will be addressed in step five, defining data requirements.

**Step Four: Develop Conceptual Model**

The conceptual model is the logic that will be later developed in computer code. Anything that can be modeled conceptually can in theory be simulated. After defining the objective and the scope develop a flow diagram of an entity through the system and use the flow diagram to consider all paths. Make sure that the scope of the model defined earlier is not being exceeded. Does the flow diagram answer the objective of the model and has the scope been exceeded? Consider these questions while developing the plans for simulation model.

**Step Five: Define Data Requirements**

A simulation model is only as good as the data that feeds it and the results are only as good as the data that is reported. So the results are only as good as the data. This step in the phase is not about collecting and data mining, but documenting what "common" data is needed for input and outputs. The third phase (Section 5) will consider the actual data. This step is merely trying to find out what data will be needed to complete the flow diagram and answer the intended question of the model. "Common" data is data that is universal to all similar systems. It is important to document the data input/output with common fields. This will make the data generic in its naming. While noting what data is needed it will become apparent what data is not readily available or difficult to find. The earlier this is recognized the more beneficial it is to the development of the simulation model. Resources can be assigned at an earlier stage in development to retrieve data that is not readily available.

**Step Six: Construct Reports**

After determining which data is required for output it is time to formulate what kind of "common" reports to create. Which reports will help to answer the objective of the model? During this step only consider what to report with the raw data that is output from the model. These "common" reports will be created in the third and fourth phase (Section 5 and 6 respectively).

**Step Seven: Brainstorm**

Finally, brainstorm other instances where the intended simulation model can be utilized. This will allow for everyone in the group to think of alternative ways that the simulation model can be used and thus will get people thinking about a "re-usable" flexible data-driven simulation model.

In summary the planning phase should be used to formulate a base plan with all interested parties. This will conclude with the development of an model objective, scope of the model, and a conceptual model flow diagram. With the conceptual model in place identify potential output/reports and identify data needed. To make the conceptual model flexible put generic terms in the definition of the diagram nodes. For example aircraft, tanks, cars, and other vehicles can be modeled the same, where all are acted upon in a simulation. Thus, when defining them in a flow diagram use a generic term such as vehicle or entity. Define what question to answer with the simulation generically by trying to generalize the reports such as availability of the vehicle. More detail will be given to this in the data structure phase (Section 5). Next, identify data needed. Depending on the type of simulation the actual data is going to be different between different entities, but the type of data is going to be similar. A mean time between failure (MTBF) and mean time to repair (MTTR) will be different in the actual data only. But the fields required for this data should be "common" between similar systems. It is just data and this can be populated at a later time to make the simulation specific. Data is tough, but don't let that stop development. Utilize fake data when real data is not available and when data becomes available simply populate the data in the simulation. The most important aspect is to analyze what "type" of data is needed. If the data needed is not available during this phase devise a plan to get the data required to model.

The goal of this phase is to answer the question, "How can I make this flexible enough to achieve the goal of creating a generic data-driven simulation model?" Answering this question through each step of the planning phase with discussion and questions is beneficial to the rest of the development.

## 4 SOFTWARE SELECTION

The next phase is to select what tools to use to develop the simulation model created in the planning phase. There are three main software component choices to make. The first two choices are necessary, while the third choice adds more flexibility. The first choice is the simulation language. Whether it is a computer off-the-shelf (COTS) simulation language or another common computer language there are a number of choices to consider. Next, step is to choose a software program for data input/output. There are number of different programs to choose from, but it will ultimately come down to a database or spreadsheet type of software. The choice of software for data is going to ultimately decipher a language for an interface. The interface is important for the user to easily change data, which is the foundation of the generic data-driven simulation. While making these important choices there are a number of things to keep in mind, which are discussed in the following steps.

**Step One: Simulation Language Selection**

The first step in software selection is deciding what type of simulation language to use. When choosing a simulation language consider the ease of use both for the developer and the end-user. The ease of use is relative to the experience of both parties. Another item of concern is an issue that was addressed in papers by Overstreet, Nance, and Balci (2002) and Paul and Taylor (2002) where models being developed for "re-use" should utilize common libraries or sub-models. In addition to this it is advantageous to de-

velop common code hierarchial. This method of coding will allow for ease of deletion, insertion, or change within simulation code. This type of development will allow for greater flexibility of the simulation. Next, consider the end user and how they will run the simulation. When selecting a simulation language select one that has a separate run time version or the ability to create an executable file. This will enable the end user to run the simulation and not have to purchase a licensed version of the simulation language. Licensing costs for developer's licenses can be very costly and must be considered in developing the model. A simulation is useless without data, so in searching for a simulation language consider the connectivity to a variety of data sources. Data connectivity is essential in data driven simulations. Consideration of what type of data source to connect the simulation to is discussed in step two.

**Step Two: Data Software Selection**

The next step in software selection is the choice of what type of software to use for data. It is important to consider when selecting a data software the ease to make changes to input data and viewing outputs of the simulation. These considerations increase the flexibility of the model for "re-use". As is the case with simulation logic consider the end-users expertise and availability of the language.

The real decision for selection of data software is going to be spreadsheet vs. database. Each has different advantages/disadvantages, but it will ultimately be decided upon what type of simulation is run. As a general rule if the simulation is producing large amounts of data and the data will be analyzed by external code (i.e. VBA, C#,…) it will be best to utilize some form of database. Spreadsheets are limited in the amount of data they contain and are slow when running macros/formulas. Spreadsheets are easy to use and many developers and end users will know how to manipulate the output data to perform analysis directly. Because of the ease of use, spreadsheets are often selected. Caution must be taken when using spreadsheets because the formulas can be broken easily and simulation runs can be compromised.

As a result spreadsheet applications should be utilized if there is minimal amount of data, formulas and code in the spreadsheet can be hidden or locked, and data output from the simulation is not raw and is more revised. Databases should be used in the case where there is a lot of data, the data output is not rolled-up, external or imbedded language are used to control access of input/output data, and ready made queries can be used to generate reports.

**Step Three: Interface Selection**

The final thing to consider when selecting software for a data-driven simulation model developed for "re-use" is a language to develop an interface. The simulation can be ran without the interface, but adding an interface to the simulation tool is advantageous for "re-use". The flexibility of a simulation is reliant on the easy of changing the input data and the simplicity in getting results of the simulation. Consider an interface language that can be easily attached to the data source.

The selection of software is necessary in the development of a generic data-driven simulation model. For each decision of which simulation language to choose, ask "How can I make this flexible enough to achieve the goal of creating a generic data-driven simulation model?" Choose a simulation language that can be run on multiple computers either using a run-time version or creating an executable version. Make sure that the simulation can easily be connected to a data software element for input/output. Finally, when making the decision on data software make it easy for the user to alter data input and view output reports. This final consideration is important in developing a generic data-driven simulation model. The more difficult it is to change the data, the more complex the "re-use" becomes. So consider a user interface with the ability to create interfaces for data input and reports.

## 5 DATA STRUCTURE

The previous sections have established a baseline for the creation of the simulation model. This phase is going to provide insight into an extremely important step of the generic data-driven simulation model. The key word in this type of simulation model is "data". The ultimate goal is to create a data structure

that will allow the end-user to easily change data inputs and perform analysis on the end result while keeping the simulation logic constant. The language or application of data input/output has already been decided. This phase will detail how to arrange the data in a logical manner so that data can easily be inserted, deleted, and/or changed.

**Step One: Data Normalization**

The first step in structuring data is through data normalization. Normalization is a way to organize data to eliminate anomalies associated with insertion, updating, and deletion. Data normalization eliminates duplicate data in tables and enables the creation of new tables with parent and child keys from other tables. This permits displaying of like information, but without the loss of data integrity when updates, deletion, and/or insertion are required (Pratt and Adamski 2005). An example of this, is an instance of taking a simulation model developed for an aircraft and altering that model to handle tanks (Figure 2). Provided the data is normalized the name changes will only need to take place in one table. In this table (vehicle) each type of vehicle is given an identification number and a representative name for the vehicle. For each vehicle type it is only necessary to change the vehicle field, which defines what type of vehicle each identification number stands for. This change will cascade throughout the data as input and all reports that are also tied to this id will now display information based on tanks as opposed to aircraft. Anywhere in the data where this Vehicle ID is referenced the data pertaining to a specific vehicle will need to be changed to correspond to the new vehicle.

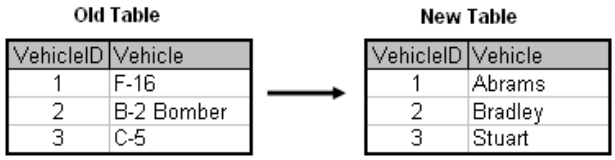


Figure 2: Altering Normalized Data Table

The idea of normalization is a very basic database concept and technique, but this technique can be utilized in spreadsheets in the same way. Create separate tabs where each tab is used for a reference table. The tables in Figure 2 are reference tables. For example, when identifying a vehicle in other tables utilize the VehicleID (Parent Key) to represent a certain type of vehicle.

Structuring data will pay off in the development of the simulation, particularly in the logic construction. The logic will be developed around the data that has been constructed in a concise manner. Spending the time on data structure and creating all tables loaded with data before developing the simulation will achieve a smooth simulation code. The alternative is a patchwork of different data types and structure with the code reflective of this type of structure.

**Step Two: Structuring Model Switches/Gates**

The next step in data structure is to plan for areas of the model to turn on/off for different systems. These are controlled by gates/switches and are utilized through binary code or by zeroing out activity delays. A gate/switch is an area in the simulation that is used to skip logic not required. These gate/switches are controlled through data. The next section (Simulation Development) will describe this to a little more detail, but it is necessary to think about these gates/switches when structuring data.

Take for example there is a delay for pre-flight inspection where a crew chief must be seized prior to performing the inspection. But, the model is now being modified to model a tank where there is no pre-flight inspection. To alter the data simply zero out the data for the delay associated with the pre inspection. To handle the issue of constrained resources, crew chief, make the resource unconstrained by assigning them to a 24/7 shift schedule and assign some arbitrarily high number available. This method

will not alter any of the logic and no delays will be associated with this activity so even though the model might go through this part of the logic no additional time is added to the simulation clock. This is an example of how a section of a model not being utilized in the current simulation can be by-passed through data without altering the actual simulation code.

The other method is to create a table made of 0 and 1s that are used as gates. Make sure that each gate is labeled with an identification number or nomenclature to identify the location of the gate. The preceding examples are some methods to think about in data structuring prior to simulation development because these will assist in model flexibility.

### Step Three: Structure Output

Finally, when structuring data it is important to not overlook structuring the output data from the simulation. When structuring the data consider the reports and objectivity defined in the planning phase. These output tables should be structured to incorporate a field to record replication number. This paper will not go over how to analyze data, but when outputting raw data from the model it is important to classify each output with a replication number.

As stated by Mackulak et al. (1998), "…a model is identified as generic, when it is applicable over some large set of systems…" and "…becomes specific when the data for a particular system is loaded". Data structuring is the foundation to developing a flexible generic data-driven simulation model because the coding of the simulation is going to be based around the structure of the data. Being able to insert, delete, and change data in one area of the data saves time and eliminates the loss of data integrity.

The data structure phase is one step that should harvest the most attention. It is this phase that will define a data-driven generic simulation. "How can I make this flexible enough to achieve the goal of creating a generic data-driven simulation model?" Utilizing thorough structuring of data through normalizing, reference tables, and data that can handle switches/gates by either zeroing out data or binary code will allow for flexible simulation model development.

## 6 SIMULATION DEVELOPMENT

The objective of planning, software selection, and data structuring is to put into place all the elements required to build a simulation model specifically designed for "re-use". The final phase of the process is to code the conceptual model developed into a computer representative model. By performing all the preceding phases the development should be much simpler than just starting out with coding of the simulation. Without getting into specific simulation languages the following are some tips to utilize when coding.

### Tip One: Code Hierarchical

The most important idea to take from this section is to code hierarchal in sub-models. Meaning separate language into common sections of code and separate this logic from other logic utilizing entity triggered gates. Entity triggered gates are sections of code in the simulation that are utilized as switches for by-passing simulation logic (Section 5.2). This will allow for ease of implementing new logic if required. In coding this way parts of the logic can be by-passed if not required, through data structuring.

Code developed hierarchically can also be used in other simulation development through use of a library of code. Utilizing a library of code is a way to save development time for other systems that might have similar logic, but are unlike.

### Tip Two: Eliminate Hard Codes

Another general rule is to use data to run the model and eliminate any if possible hard codes. Hard codes are those codes or data that are inherently coded in the logic and can only be altered through simulation code. If the choice is to put a standard hard coded delay in an activity and the use of data to create

the delay, choose the later. The later is flexible and is easily altered. As opposed to hard coding a delay which is not flexible and changes can be hard to find.

**Tip Three: Code for Expandability**

There will be instances where code can be included in the model, but under current conditions will not be modeled. This is not always the case for expandability, but often times it can be determined where new code could be added. In this instance create logic to model for expandability. If while coding there is a consensus that at some juncture in the model new logic might be added, route an entity to the new logic. The entity does not have to be sent through any logic, but plan for where the new logic might be implemented. This prevents finding where to insert the new logic at later period of time if needed and speeds up development.

**Tip Four: Output Common Data Elements**

Finally, consider the objective of the simulation and the "common" reports discussed in the planning phase. Then looking at the structure of the output data identified create logic to output this data in the form it was structured. These will be "common" data elements that will be used to formulate final reports, which will answer the overall objectivity of the simulation.

The development of the simulation/code is not as profound as the other phases of generic simulation development, but consider the steps in this phase while coding. Important to note in coding the simulation is to consider developing hierarchical, eliminate hard-coding whenever possible, and plan for expansion as ways to create a flexible generic data driven simulation model.

## 7 CONCLUSION

The development and maintainability of simulation is time consuming and as a result is expensive. But, what if there was a way to create one flexible model that was data driven and built for "re-use" of similar systems? Then what if because of its "re-use" the costs over its many uses on multiple similar systems is reduced? This is something that management might consider when performing a cost benefit analysis of the development. This paper was created using the techniques used in current model building, but with the caveat of expanding the development to consider the model flexibility. The ideas and the methods of which developing simulations are largely based on the steps in Simulation with Arena (Kelton, Sadowski, and Sturrock 2004). Using these steps as a baseline for development and then expanding them to create flexible generic simulations is what sets these phases apart from the standard development practices. Generally, the next phases are validation and verification. These have been purposely left out of this paper and is the baseline for future research. In general planning is the foundation for developing a generic simulation.

First, develop the conceptual model and decide which data will be needed. Sometimes the development of the code of the simulation becomes the most important and we lose track of what is trying to be answered through the simulation. So know why the simulation is being developed and as a result the answer to the question will help in deciding what data to output and what reports to develop. If it's the intention to build a model for universal uses this is the point to brainstorm additional uses for the simulation. These are important in phases later on.

Select the software to use for both data input/output as well as the simulation language. When making the selection be sure to consider run-time versions of the language for COTS packages and ease of use for the end-user.

Create an easy flexible data structure that can be altered to easily change between different scenarios as well as different modeling capabilities. Utilize common database techniques such as normalization and reference tables. These techniques are the foundation of creating databases, but with some ingenuity can be developed in spreadsheet software (with some code) as well.

Finally, when the entire framework is laid out and everyone involved is on equal footing it is time to develop the code for the simulation. The coding of the simulation is an art and no one method is wrong, but there are some ways to make development more user friendly. Develop hierarchial, attempt to code without any hard-codes, and plan for expansion. Developing hierarchical allows for ease of implementing and/or deleting logic. Then utilizing gating data and logic certain blocks of logic can be by-passed. Hard-coding will immediately make your model inflexible and the more inflexible the model the more effort it will take to implement the simulation to other applications.

The preceding ideas should be catered to specific cases and are generally important phases to follow whether it is a one time use or multi-use simulation model. These phases will benefit the developers in either case because they are heavily focused on planning prior to coding the simulation logic. As developers we tend to focus on the end product and do not plan accordingly. The ideas of this paper are trivial and the ideas are nothing more then planning as opposed to going directly to the end product. As a result of the planning, developers can begin to start developing data-driven simulations that might see the light of day in other capacities as opposed to sitting on a shelf collecting dust after the original question has been answered.

Many papers have been written on this topic both positive and negative. While there have been many papers on the topic there was no one paper found that attempts to lay down the framework for how to develop these types of models. This is where this paper differs and these ideas will continue to grow as is the case with many simulation models, but planning for them allows for expansion.

## REFERENCES


Brown, N. and S. Powers. 2000. Simulation in a Box (A Generic Reusuable Maintenance Model). In *Proceedings of the 2000 Winter Simulation Conference,* ed. J.A. Jones, R.R. Barton, K. Kang, and P.A. Fishwick, 1050-1056. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc..

Kelton, W.D., D.A. Sadowski, and D.T. Sturrock. 2004. *Simulation With Arena.,* 3rd ed. New York, New York: McGraw Hill.

Mackulak, G.T., F.P. Lawrence, and T. Colvin. 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. Manivannam, 979-984. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Overstreet, C. M., R. E. Nance and O. Balci. 2002. Issues in Enhancing Model Reuse. In *First International Conference on Grand Challenges for Modeling and Simulation,* ed. W.H. Lunceford and E.H., 1-5.

Paul, Ray J. and Simon J.E. Taylor. 2002. What Use is Model Reuse: Is There a Crook at the End of the Rainbow?. In *Proceedings of the 2002 Winter Simulation Conference,* ed. E. Yucesan, C.H. Chen, J.L. Snowdon, and J.M. Charnes, 648-652. Piscataway, NJ: Institute of Electrical and Electronics Engineers, Inc.

Pratt, Philip J. and Joseph J. Adamski. 2005. *Concepts of Database Management,* 5th ed. Boston, Massachusetts: Thomson.


## AUTHOR BIOGRAPHIES


**NICK BROWN** is the Senior Engineer responsible for modeling and simulation development. He holds a M.S. in Human Factors Engineering from Wright State University and a B.S. in Business Administration specializing in Marketing and Logistics from the Ohio State University. His thesis worked included simulation work to evaluate his inventory recommendations and he has been developing simulation for KLSS for 3+ years with emphasis on availability and sustainment. His email address is <nick.brown@klssinc.com>