Proceedings of the 1992 Winter Simulation Conference
ed. J. J. Swain, D. Goldsman, R. C. Crain, and J. R. Wilson

MODEL REUSABILITY IN A GRAPHICAL SIMULATION PACKAGE

Betty J. Bortscheller
Emilie T. Saulnier

GE Research and Development,
Schenectady, NY 12301, U.S.A.

ABSTRACT

Model reusability is becoming more important as simulations grow larger and more complex. Although graphical user interfaces have already proved to be a valuable tool in many phases of discrete event modeling, simulation, and analysis of computer networks, their potential for supporting model reuse has not yet been fully exploited. This is partly because although several graphically-based commercial packages are available of varying levels of maturity, none yet provides all of the features needed to effectively reuse simulation models. In this paper we describe the general requirements needed to support model reusability in the "ideal" graphical discrete event simulation package. The BONeS simulation package is used as one example of how a commercially available graphical simulation package currently supports these requirements to various degrees. The most critical areas for improvement are discussed.

1 INTRODUCTION

With the advent of large and complex systems such as distributed communications networks, simulations are becoming proportionately large and complex. This increasing complexity mandates cost-effective, integrated and automated support of simulation model development throughout the entire model development life cycle (Balci et al. 1990).

Since simulation is essentially software, it is natural to try to apply software engineering techniques such as object oriented design, top-down design, and bottom-up validation to improve simulation efficiency and reliability. For example, many of the computer aided design techniques that were used previously for hardware and software design have been successfully applied to system simulation (Jerome et al. 1987, LaRue et al. 1989, Saulnier et al. 1988). In fact, several commercial packages have emerged (including BONeS, SES/workbench, OPNET, LANSIM), all of which support discrete event simulation through a graphical interface.

One design technique that has yet to live up to its promise in either the software engineering or simulation fields is that of model reusability. This technique is particularly well–suited to simulation because of the iterate-and-refine nature of simulation experiments. However, as has been found in software engineering, reusability cannot occur without an organizational infrastructure which supports it (Priéto-Díaz 1991). This infrastructure includes support for development, validation, documentation and maintenance of reusable models, an easy-to-use library system, and reuser support (see Figure 1).

Much of the reuse infrastructure that is needed could be provided by a graphical simulation package. Unfortunately many of the packages currently available are very strong in graphical capture of design but provide minimal, if any, support for model reuse. In this paper we introduce a framework of requirements needed to support model reusability in an integrated graphical simulation environment. We use the BONeS (Block Oriented Network Simulator) (Comdisco Systems Inc. 1992) simulation package to illustrate how one commercially available tool supports these requirements to varying degrees. Finally, we summarize the most critical areas for improvement.

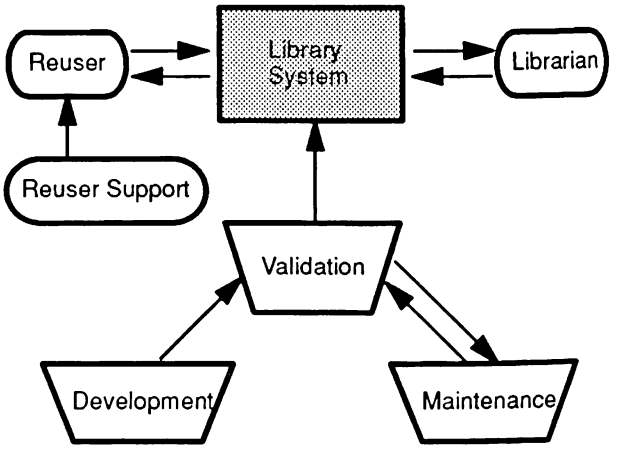

Figure 1: Organizational Infrastructure Required for Model Reuse

2 WHY A GRAPHICAL ENVIRONMENT?

Although a reusability infrastructure could certainly be provided without a graphical environment, there are many advantages to using a graphical tool to support model reusability. These advantages are summarized below.

Model Development: Perhaps the most important advantage is that a graphical simulation tool provides the user with an integrated environment that allows the user to focus on the high-level details of the model and simulation experiments. Not only can it insulate the user from the programming language details, it can also free the user from knowing the commands, file structure, and other specifics of the operating system. As a result more time can be focused on careful specification, design and implementation of a reusable model, and even facilitate adoption of proposed frameworks for design-for-reusability (Pratt et al. 1991, Harel 1992).

Model Validation and Documentation: Another benefit is that a graphical simulation tool by its very nature provides a graphical representation of the model. This graphical representation is essentially a form of self-documentation and simplifies the task of understanding the logical design of a model. This graphical representation of the model and related results also simplifies the process of design reviews and debugging that is essential for verification and validation. Naturally, the potential reuse of a model is easier to assess when its behavior is easy to understand and verify.

Model Library and Maintenance: A good graphical tool can also automate software and project management. Requirements include not only configuration and version control, but also model categorization and search functions. Documentation functions are also needed for effective model reuse.

3 GENERAL REQUIREMENTS

Model development requires the construction of a model that represents all of the "important" aspects of the system (Pollacia 1989). The key purpose of the graphical environment is to free the developer from inappropriate levels of detail and thus simplify the translation of an idea for solving the problem into an appropriate high-level model specification (Harel 1992). For example in the modeling and simulation process proposed by Garzia (1990), the addition of a well-designed graphical interface can simplify the transformation from a conceptual to a computational model, automate much of the implementation phase, and aid in verification and validation (see Figure 2). The general requirements include user friendliness of the environment, the clarity of the graphical depiction, and support of the specific requirements of the modeling and simulation process.

3.1 User Friendliness

In order to be accessible to the widest range of users (both model developers and potential re-users), the user interface needs to be easy to use. It is preferable that standard use be made of the mouse, windows, and keyboard. Our example tool, BONeS, runs in X-windows and incorporates the graphical user interface look-and-feel called OSF/Motif. Therefore the user interacts with BONeS in a way consistent with other Motif-based interfaces.

Using the interface look-and-feel as a foundation, the user's choices should be organized and conveyed so that the next step a user needs to perform should be easily accessible and almost intuitive. A rich set of capabilities will also encourage use of the tool. For example, the editing capabilities should include not only basic features such as move, copy, and undo, but also "convenience" features such as replace and group-edit.

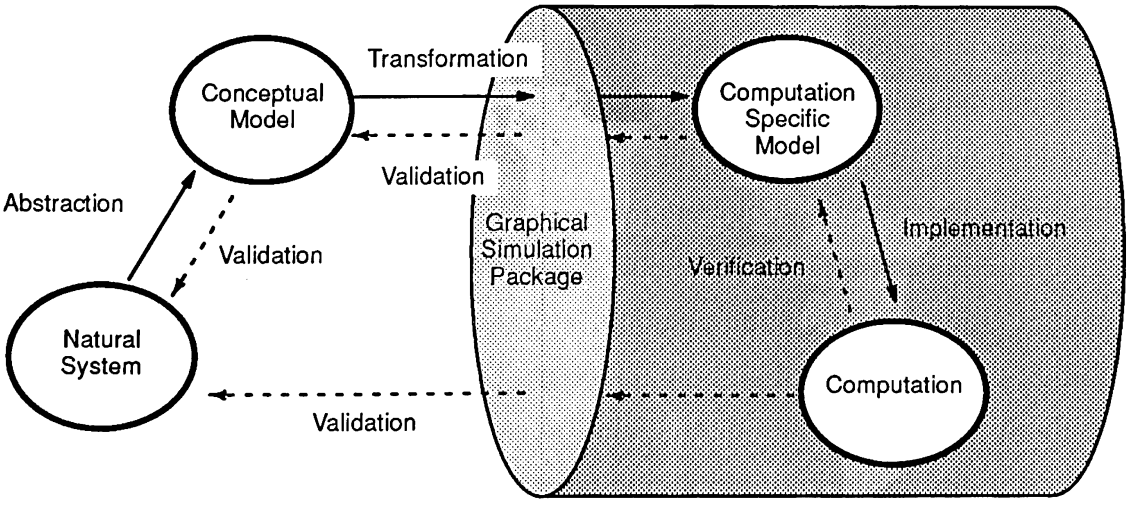

Figure 2: Role of the Graphical Interface in the Modeling Process

The learning curve can be further shortened if the interface is context sensitive and does not allow the user to perform inappropriate actions. This guidance should include **helpful** error messages and prompting of appropriate actions when there is an error, as well as on-line help. Our example tool, BONeS, offers error messages when an inappropriate action is made by the user and offers some information on what the user should do to correct the situation. BONeS also offers detailed on-line help with the user being able to select help on a specific part of an object or the entire object. Although this type of user-friendly design should seem obvious, many tools still are inadequate in this area.

3.2 Clarity of Graphical Depiction

Clarity of graphical depiction is important even for a non-user. For example, design reviews require documentation that can be easily explained and reviewed for logical errors by non-users with a variety of backgrounds. Similarly, a reuser needs to be able to quickly understand the model. Diagrams that don't clearly depict the model can be easily misunderstood and result in misuse or replication of models.

The clarity of a graphical depiction is affected by the design paradigm, the iconic representation, and labeling capabilities. For example, BONeS employs a data flow design paradigm so that a model diagram consists of blocks with connections between the blocks that represent the flow of data in the model. The developer can assign names to blocks that reflect or clarify their function and basic annotation capabilities are provided. This helps even the non-user to understand the model function. One shortcoming is the lack of user defined model i/o labels, which becomes even more important in reuse of higher level models that don't have simple in to out data flow.

Another shortcoming is the limited (although improving) capability provided for the user to define the icon representation of a model. This capability is essential to a clear graphical depiction: the power of a graphical representation is not being fully utilized unless the appearance of the blocks in the diagram helps convey their function. This capability is complicated in a general simulation package since each different application often requires different, and sometimes unique, depictions. However, the crucial first steps of the modeling process will be poorly supported as long as users have to first use pencil and paper (or another graphing package) to design or explain their model.

3.3 Support of the Modeling, Simulation and Analysis Process

Of course the process of modeling, simulation and analysis has many specific requirements beyond that of the generic graphical capture user interface. Figure 3 illustrates the simulation model development process in terms of a fountain life-cycle previously proposed for object-oriented systems development (Henderson-Sellers 1990). The impact of each phase of this life-cycle on the graphical simulation package requirements as they pertain to reusability will be described in the rest of this paper.

4 MODEL SPECIFICATION

Model specification requires both functional specification and parameterization.

4.1 Functional Specification

Top down design greatly enhances functional specification, especially in a teamwork situation where interfaces must be defined first. Top down design allows incomplete models to be used to define high level interfaces. This allows one group to continue progress on the model while another group is ironing out bugs in their particular portion of the model. Of course a simulation cannot be run unless all models are completed. This is one reason that tools such as BONeS do not normally support top down design (although BONeS does provide a save-incomplete that allows models with only inputs and outputs to be defined). The facilitation of top-down design would allow more careful interface design and thus enhance model reuse.

Hierarchy is also a valuable tool in design specification. Hierarchical design allows a large complex model design to be broken into several smaller functions that can be individually developed and specified. This method is often described as "divide and conquer," and is similar to the use of subprograms in traditional programming languages (Jain 1991). The hierarchical modeling approach is especially valuable when used to manage modeling complexity and to capture the layered architectures of networks. As in top down design, hierarchy can be used to define interfaces for and isolate reusable parts of a system

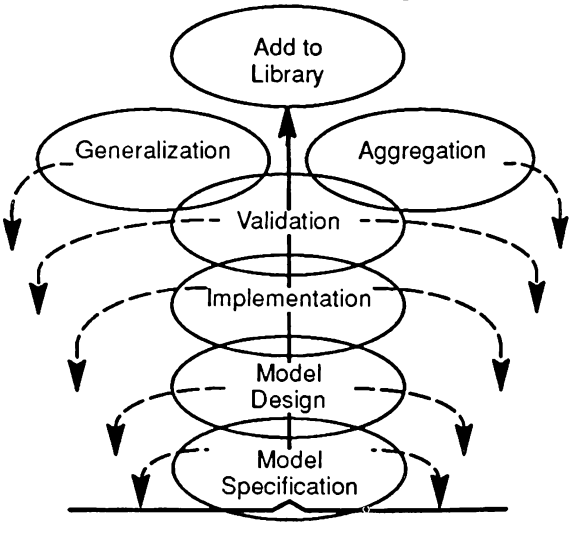

Figure 3: The Fountain Life-cycle Model for Object Oriented Software Design as Applied to Simulation

specification. See Figure 4 for an example of an hierarchical block diagrams in BONeS.

An especially valuable feature which is not yet available is the ability to define hierarchy in a top down manner. In this way a logically coherent parts of a large complex diagram could be "reduced" to a submodel in an automated fashion. This feature would accommodate the common practice of designing and verifying a simple model upon which additional complexity is added as need. Without a "reduce" feature this leads to large crowded diagrams that are impossible to follow.

4.2 Parameterization

Parameterization is a fundamental requirement for trade-off analysis simulations. The flexibility of parameter specification in a graphical simulation tool is even more important in the context of model reuse. Models used for similar purposes often contain similar components although the parameterization of these components may differ (Cellier et al. 1990). If models can be customized for different applications by parameter specification they are inherently more useful.

Parameterization of a reusable model is simplified if the graphical simulation package checks for type consistency so that inappropriate types can not be assigned as parameter values. The model developer should also be able to specify ranges to prevent inappropriate specification by the reuser. This feature is especially important when the model reuser is not the original developer.

Similarly, a parameter should be able to be defined in terms of other parameters. In other words instead of assigning an explicit value to a parameter the user should be able to assign a value based on other parameters. By using these types of expressions for parameter values, it reduces the number of model (external) parameters and simplifies the process of assigning values when the block is used at a higher level. This approach also prevents parameterization errors by allowing any assumed relationships between parameters to be hard-wired into the model.

Finally, a graphical simulation package should allow for parameters to be assigned at the appropriate hierarchical level in the model. Certainly, parameters that have a fixed value regardless of simulation goals would be best assigned at the lowest hierarchical level. In this way the user can defer assigning a value only to parameters that will be needed for trade-off analysis at the highest level in the hierarchical model.

The BONeS simulation package provides a full range of parameter support. It does type and range checking on all parameters and allows the user to define parameters in terms of other parameters. BONeS also allows the user to assign values to parameters at the hierarchical level in which they first appear or to defer assigning a value to a

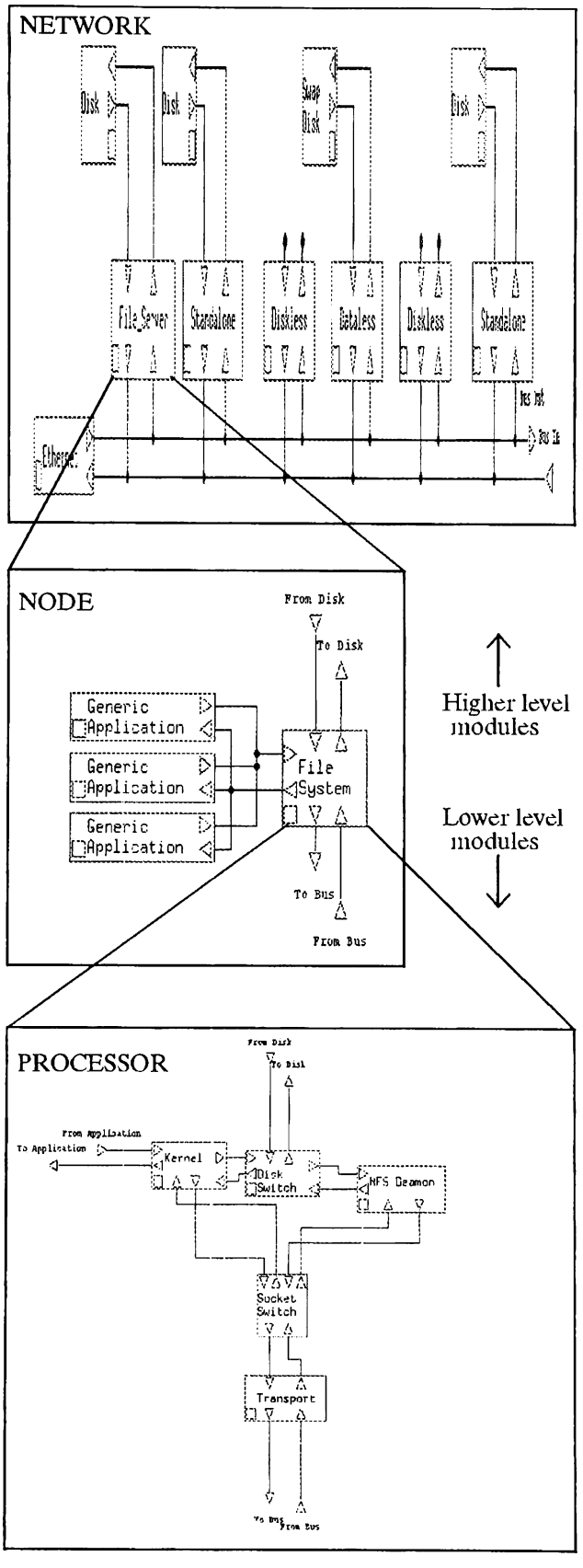

Figure 4: A BONeS Hierarchical LAN Model

higher level by exporting the parameter. Finally, at the simulation level BONeS allows iterated parameters. In all, this area is well covered by at least the BONeS package.

5 MODEL DESIGN AND IMPLEMENTATION

In a graphical simulation package, once a model is specified it is designed and implemented using what we will call building blocks. These building blocks can also be thought of as reusable models.

In any graphical simulation package there are two basic types of building blocks; primitives and higher level. Primitives are the lowest level model and must be specified using a programming language, while higher level building blocks consist of one or more interconnected primitives. A user can avoid introducing programming errors by avoiding design of new primitives and specifying only higher level models.

Additionally, a particular building block may be either pre-defined or user-defined. Pre-defined building blocks are reusable models supplied by the vendor. They should be generic in nature and comprehensive enough to encompass many modeling possibilities. User-defined blocks are developed by the user and provide a way to further extend and customize the tool capabilities either at the primitive level or at the higher level. Of course the more flexible the pre-defined building blocks are, the fewer user-defined blocks will have to be created. It should be transparent to the user whether a given building block is pre-defined or user-defined, or a primitive or higher level module. Any module should be able to be reused in any other module.

For example, BONeS supplies a user-extendable library of building blocks. The library consists of approximately 200 models. The users can use these models to create their own models and then add the user-created models to the BONeS library. BONeS also provides the user with the capability to create primitive models using the C programming language and to add these primitives to the user's BONeS library. In addition, BONeS provides specialized libraries such as a LAN model library.

Of course, the more general a model is the more it can be reused. In order to support this generality the tool must also provide a method for inputs and outputs to be data type independent so the module can accept different data types. In the case of BONeS there is a **deferred** data type that allows the user to create models that are not data type specific so they can easily be reused.

6 MODEL VERIFICATION AND VALIDATION

Once a model is designed and implemented, the next step is usually verification or validation through simulation. In order to support this phase of the process, a graphical simulation tool must support simulation specification, control, debugging, and data collection and processing. The capabilities a tool provides in these areas not only impacts the reusability of simulation models during a given experiment, but also allows more thorough testing of reusable models.

6.1 Simulation Specification and Control

The capability to easily run parameter iteration simulations and run confidence interval calculations is critical. Parameter iterations allow the user to efficiently run many simulations while gathering the necessary statistics on each iteration for trade off analysis. Similarly, multi-batch simulations can simplify the use of confidence intervals to measure the statistical significance of a simulation result. The impact on reusability is simplified (and therefore hopefully more thorough) validation and model characterization. The BONeS package provides simple specifications for both parameter iterations (over a set of available hosts) and confidence interval calculations.

The ability to control the simulation is also important, although this area has less to do with model reusability than to do with usability. For example, changing the priority of a simulation allows better use of a workstation CPU and may allow the user to borrow "extra" time on other machines. Similarly, ability to pause and restart a simulation is more efficient than killing and restarting. Finally, availability of data and statistics on-the-fly allows the user to monitor simulation results as they are collected. BONeS does allow the user to pause and restart simulations. Viewing of intermediate data results is also available but should also be extended to wrap-up statistics.

Another useful feature is to use an assortment of triggers, other than simulation time, to stop a simulation run. The user should be able to stop a simulation run based on a parameter reaching a specified value. If the user can pause a simulation to check values of interest, it may prevent aborting a time investment in an already long simulation task. This feature should also include the option of stopping a simulation before it is actually finished and modifying simulation parameters (e.g. to insert failure points). In the BONeS package a stop point for a simulation is set by the user specifying a stop time or specifying a parameter value to stop simulation at. This capability really needs to be expanded to be truly useful, say run until a given confidence interval is reached. It is also not yet possible to adjust parameters during simulation runs.

Finally, many times a user interface host is not the most efficient place to run long compute intensive simulations. The tool should allow the user to compile and remotely run the simulation on other hosts that may be available. Similarly, windowing support is not always available at all terminals. It would be helpful to have some basic simulation specification capability available without invoking

the graphical interface. The BONeS simulation manager allows the user to specify which processor the simulation is to execute on and to set a priority for the simulation.

6.2 Data Collection and Statistics

The results that are needed in a particular simulation may vary and are closely tied to the problem being investigated. The simulation tool should be able to eliminate the need for a hand-held calculator as a post processor! In addition, since the collection of data often impacts the length of the simulation, it is important that the simulation tool allow for flexibility in specifying the data that is collected and the method for collecting. For example, some tools allow the user to define which types of statistics to collect and other tools automatically collect certain statistics such as mean, variance, minimum, and maximum. For example, flexibility is becoming even more important as new statistical approximations are used to characterize correlated network traffic.

Another important issue is the location in the model at which the data collection takes place. For example, some tools allow the user to specify the location of data collection and other tools automatically collect data at all modules. The ability to be selective can have a profound effect on CPU time needed to run the simulation as well as the CPU time needed to plot the results.

Other options for statistical collection should include the length of time to collect the data for and whether to calculate on the fly. Of course sampled data should always be an option. The ability to write the data collected during simulation, or a portion of it, to a file gives the user another method for debugging and verifying the model.

The BONeS package applies a paradigm from the hardware world by allowing the user to choose where data is to be collected by placing a "probe" at the appropriate input or output. BONeS offers different types of probes including generic (sampled data), batch mean, batch statistics, histogram, and throughput and delay. In addition users can create their own probes for specific statistical collection and add them to the BONeS probe menu. When a probe is placed on a BONeS simulation, the user is informed as to what data is available at that probe location and is prompted to select the specific data to be saved during simulation.

6.3 Model Testing

Although model testing is important in any simulation project, it is of even greater importance if a model is to be reused. Since simulation is essentially software, some testing support is needed to verify simulation operation in the face of unexpected results. A simple version of testing is a simulation trace at various levels. Animation can often serve as an excellent testing tool for locating the spot in the model where the problem is occurring. Finally, a graphical simulation package that does consistency checking of the model can prevent a lot of errors that a user would spend time trying to locate and correct.

A related issue is that of a test suite for reusable models. Each reusable model really needs an accompanying test program that can be run and compared to the original results. In the absence of this type of semi-automated test, a reuser will eventually have to spend a significant amount of time testing each reused model. This issue is especially important to detect bugs related to the "unanticipated consequences" of software updates.

Testing support is one area that needs improvement in all currently available tools. This is one area where capabilities commonly available for software debugging have not been transitioned to the simulation environment. For example, BONeS does not have a mechanism for providing the user with a simulation trace. Although BONeS does have animation, it is playback animation only and is not interactive.

In fact, debugging support is probably the most critical area where the BONeS package, specifically, could be improved to promote model reuse. There appears to be a large void in this area of the tool in comparison to the other capabilities. Although not directly related to reusability the lack of debugging support can seriously impact the quantity and quality of models developed for reuse.

6.4 Trace-driven Simulation

As good as statistical load models are, in many cases they aren't able to adequately represent the flow of actual data. Therefore, in some cases it is advantageous to run simulations using measured or empirical data. Since the data reflects real traffic patterns, users could run multiple scenarios to find the most efficient configuration. Although it is not part of the standard BONeS package, Comdisco Systems Inc. does have an interface available that allows a BONeS model to accept empirical data from a Sniffer network monitor as simulation input.

Empirical data also has its limitations: the biggest one being the difficulty of collection. An alternative method is the use of scripts or scenarios to specify the correlation between network application traffic. This method has the additional advantage of abstracting the traffic characterization to the application level rather than the packet level characteristic of measured data.

6.5 Results Analysis and Presentation

Before a model is reused it is imperative that the model behavior be verified to avoid errors being replicated in other models. A tool that offers flexible results analysis can help the user perform a thorough verification.

In order to effectively depict the simulation results it is important that the user be able to customize the graph as needed to emphasize significant results. A flexible tool

will allow the user to adjust ranges on the axis as well as chose line types, symbols, and color for each data set The capability to display multiple plots and axes simultaneously for comparison enhances the efficiency of a tool.

Another useful feature is the capability to compare data from different simulation runs. For example, before reusing a model a reuser may want to test the model to confirm it behaves as documented. Similarly, the capability to plot theoretical or other data along with simulation data is necessary for evaluating the correctness of simulation results. Confidence intervals and other statistical tests are another critical feature.

Our example tool, BONeS, has an extensive post processor that analyzes and plots simulation results. The post processor has many features that allows the user to examine and modify the plot for important results. The tool has a documentation feature that prompts the user for input whenever new models or simulations are created or modified. BONeS provides a way of copying any BONeS window into PostScript format which can be incorporated into many text processing systems. BONeS also has a method of plotting out the actual data values from the data sets. Overall the post processor is well designed. Areas for improvement include the ability to plot expressions that use data from two simulations, and the ability to do correlation type calculations where the current data sample is compared to another in the same stream.

7 MODEL LIBRARY SYSTEM

Issues related to the model library include configuration and version control, documentation, and model categorization and search functions.

7.1 Documentation

Thorough documentation of models allows for a more informed reuse of the model. When all components of the model, such as parameters, inputs, and outputs are well documented as well as the model itself, it makes the function of the model clearer to all considering it for reuse. It also provides insight as to where modifications might be needed to better fit the model into the current application.

Since documentation, presentations and reports are a way of life, it is important to be able to print a hard copy of the model and the model documentation. Another feature that is important for reports is the ease in incorporating diagrams of the model and plots into a word processor such as interleaf. Similarly, simulation results need to be easily incorporated into written reports. Even beyond the reuse of a specific model, documentation plays a key role in being able to replicate or modify simulations for further study.

Our example tool, BONeS, has a documentation feature that prompts the user for input whenever new models or simulations are created or modified. This is useful if the developer uses it! BONeS also provides a way of copying any BONeS window into PostScript format which can be incorporated into many text processing systems. The data values of each data set can also be plotted. The biggest shortcoming is the limited annotation that is provided, and the difficulty in reading the legends of crowded plots. Support for dates and times on the print outs would also be useful.

7.2 Configuration and Version Control

In order for a large simulation model to be developed by several designers, a design management system is essential. This system is needed to maintain version and configuration control so that changes to the system may be administered by a central database.

The system should control model access to prevent multiple editing sessions from occurring to the same model concurrently and as such should provide separate working libraries for each user or group. The design management system should support read and write permissions to these libraries or models, with owner, user, and group access defined much as in a file system. The user needs to be able to set read and read/write permissions on all models created. For example, a model created for reuse could be set with read only permissions to prevent modifications being done that might cause problems with other instances of that model in use.

When a reusable model is modified, users should be alerted that the model has been changed and occurrences of that model need to be updated. It is also important that a history of revisions be provided not only to assure users that they have the most recent version available, but also to debug sudden operational differences. The tool should provide a simple automatic updating process.

The BONeS design management system provides configuration control and locking of models that are open with write permission to other users. The BONeS database manager allows the user to specify read/write permissions for owner, user, and group on every model the user creates.

Each time a new object is created in BONeS the user is prompted to set read/write permissions. Whenever a model is modified, any model that incorporates the changed model needs to be updated. BONeS has a feature to update libraries that contain modified models as long as no inputs, outputs or parameters have been deleted or added. If the modification did include the addition or deletion of those objects the user must manually update any models that are effected by the change. The utility that performs the updating provides the user with a list of the models it could not update in the specified library.

7.3 Library Organization and Retrieval

Model reuse can be simplified by a good library. If it is simpler for a user to recreate a model than to actually iden-

tify and retrieve an existing model, any other infrastructure for model reuse is wasted.

The tool needs to provide the user with the ability to establish user and project libraries to organize the models, making them easy for a reuser to locate and use.

In order for simulation models and submodels to be reused, a model library capability is essential to categorize models and provide basic information about previous verification and validation. This library must be organized logically and also provide some assistance in locating useful models. For example, BONeS takes a first step in this direction by providing a object name search function. However this function needs to be expanded to include searches of documentation text and allow searches to be narrowed by user or library type.

Reusability also requires organization in terms of user or project libraries with permissions which can be set to allow read or write access to the appropriate user or group. Model revisions should also be tracked so that a reuser can either accept model updates or find documentation as to the impact of changes. A shortcoming of BONeS, which actually has a comparatively advanced library system, is that a reuser does not have a choice of which model version to use and must always update to the most recent.

Once a reusable model is located it must be incorporated into the reuser's model. In many routine development environments, a major block to the effective reuse of existing functional models is the difficulty or impossibility of employing a model as a submodel to another model (Freeman 1983). This process can be as uncomplicated as simply referring to it like any other available model or it may involve declaring an external reference to the model. In BONeS all of the models appear together with the basic building blocks in the same set of menus. Therefore model reuse is a relatively simple task as the user uses the same method to reuse a model as to use any other model.

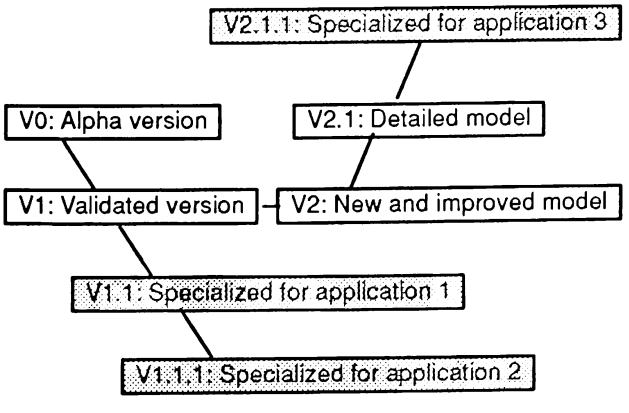

Figure 5: Revisions Should be Tracked so that Both Generic and Specialized Versions are Available to Reusers

Perhaps the biggest hurdle to reusability of simulation models is that, unlike many software programs, the level of detail required in a model may vary considerably with the experiment. Thus a fully detailed Ethernet model may be overkill for some applications, while the assumptions used for a simplified version may be inappropriate for others. As such it is likely that any reuse may involve some degree of changes and the result should be a revision still related to the original model (see Figure 5). The question of which revisions should be inherited also impacts reusability. For example, it would be desirable to inherit bug fixes without necessarily losing the custom part of the model. The closest our example package BONeS comes to providing inheritance (as used in object-oriented design) is in the specification of data structures.

The BONeS database manager is very flexible. It allows the user to define his own libraries for saving all models, simulations and plots. The user can also specify at which level in the appropriate BONeS menu the library name should appear. In addition BONeS supplies the capability for the user to group the models in a particular library into subgroups.

Although the BONeS database capability is advanced compared to many other commercial tools, there is still much room for improvement. Ideally, a reuser needs a query interface that can locate promising models in the library. Particularly challenging aspects include how to organize similar models at different levels of abstraction, and how to organize the vast amount of data that may be collected during an ongoing simulation experiment. Since the current solution in BONeS is to overwrite the previous results, a new simulation must currently be created for each new set of data collected with a slight parameter change. This solution raises some difficulties when comparing consecutive runs of the same simulation with slight changes in parameters. The parameter iteration feature solves this problem partially, but to be more effective the iteration should be based on a list rather than a loop with index.

8 CONCLUSIONS

Model reusability is becoming more important as simulations grow larger and more complex. Although graphical simulation packages are a potentially valuable tool, they have yet to support all the requirements that would make model reuse a reality.

Most of the shortcomings are reflective of the immense database requirements associated with model library organization and retrieval. Since test cases are an essential part of model documentation and validation, additional requirements are imposed on the database to store, retrieve and filter data previously obtained for library models. Finally, simulation is unlike generic software in that one real-life concept or system may have several model

representations at various levels of abstraction. Categorizing and reusing these models appropriately will require a sophisticated expert system.

An extension of model reusability is the concept of separate development and user groups. In this paradigm models are developed and validated by one group, and then used to specify simulations by another group. This concept is an advanced application of model reusability, and requires a sophisticated library and validation process.

Another issue that has yet to be addressed is the question of who will maintain reusable model libraries. In our example Comdisco Systems has developed and supported reusable models, while other vendors view model libraries as a third party responsibility. Alternatives to these approaches include development of internally maintained libraries, and informal shareware among user groups. However once model reuse is more commonplace, issues such as copyright protection will become important and will impact library development and availability.

Since simulation is by its nature an iterative process the issues of reusability affect every simulation experiment to some degree. Although simulation packages have yet to live up to their potential for providing a model reuse infrastructure, the adoption of user interface and windowing standards will shift new emphasis to support for value-added functions such as model reusability. It is expected that not only will more tools provide database capabilities similar to that of BONeS, but that these basic database and presentation capabilities will be expanded to provide the efficient documentation, storage and retrieval of simulation models, and other features needed to make model reuse a reality.

REFERENCES

Balci, O., R. E. Nance, E. J. Derrick, E. H. Page and J. L. Bishop (1990) , "Model Generation Issues In A Simulation Support Environment", In *Proceedings of the 1990 Winter Simulation Conference*, O. Balci, R. P. Sadowski and R. E. Nance, Eds. IEEE, Piscataway, NJ, 257-263.

Cellier, F.E., Q. Wang and B. P. Zeigler (1990) , "A Five Level Hierarchy For the Management of Simulation Models", In *Proceedings of the 1990 Winter Simulation Conference*, O. Balci, R. P. Sadowski and R. E. Nance, Eds. IEEE, Piscataway, NJ, 55-64.

Comdisco Systems Inc. (1992), "BONeS User's Guide ", Foster City, CA.

Freeman P. (1983), "Reusable Software Engineering: Concepts And Research Directions," In *Tutorial on Software Design Techniques*, P. Freeman and A.I. Wasserman, Eds. IEEE, Piscataway, NJ, 63-75.

Garzia, M. R. (1990), "Discrete Event Simulation Methodologies and Formalisms," *Simulation Digest*, Volume 21, No. 1., Summer, 3-13.

Harel, D. (1992), "Biting The Silver Bullet: Toward A Brighter Future For System Development," *Computer*, Volume 25, No. 1., January, 8-20.

Henderson-Sellers, B. and J. M. Edwards (1990), "The Object-Oriented Systems Life Cycle," *Communications of the ACM*, Volume 33, No. 9., September, 142-159.

Jain, R. (1991), *The Art of Computer Systems Performance Analysis: Techniques For Experimental Design, Measurement, Simulation, and Modeling* , John Wiley & Sons, Inc.

Jerome, A.C., S.P. Baraniuk, M.A. Cohen (1987), "Communication Network and Protocol Design Automation", In *Proceedings of 1987 IEEE Military Communications Conference (MILCOM'87)*, Washington D.C., October 1987, 334–339.

LaRue, W.W. , V.S. Frost, K.S. Shanmugan, E. Komp, D. Reznik, S. Schafer (1989), "A Block Oriented Paradigm for Modeling Communications Systems", TISL Technical Report TISL-8670-1, University of Kansas Center for Research, Inc., Lawrence, Kansas.

Pollacia, L. F. (1989), "A Survey of Discrete Event Simulation and State-Of-The-Art Discrete Event Languages," *Simulation Digest*, Volume 20, No. 3., Fall, 8-25.

Pratt, D. B. , P.A. Farrington, C.B. Basnet, H. C. Bhuskute, M.Kamath and J.H. Mize (1991), "A Framework for Highly Reusable Simulation Modeling: Separating Physical, Information, and Control Elements,"In *Proceedings of the 24th Annual Simulation Symposium*, April 1991, 254-261.

Priéto-Díaz, R. (1991), "Implementing Faceted Classification for Software Reuse," *Communications of The ACM*, Volume 34, No. 5., May, 88–99.

Saulnier, E.T., B.J. Bortscheller, B. A. Hamill, J.A. Langan, R. A. Locke and B.J. Scholz(1988), "NADE: Network Assessment and Design Environment: User's Guide," GE Internal Report, GE Research and Development Center, Schenectady, NY.

AUTHOR BIOGRAPHIES

BETTY J. BORTSCHELLER is employed at General Electric's Research and Development Center. She received a B.S. in computer science from Union College. Her research interests are focused on the modeling and simulation of data communication networks.

EMILIE T. SAULNIER received her B.S. and M.S. degrees from Rensselaer Polytechnic Institute, where she is currently pursuing a PhD in the area of network traffic characterization. She joined GE Corporate Research and Development in 1986 and is currently technical coordinator for the networks research projects.