Issues in Enhancing Model Reuse

C. Michael Overstreet
Computer Science Department
Old Dominion University
Norfolk, VA 23452-0162, USA

Richard E. Nance
Osman Balci
Department of Computer Science
Virginia Tech
Blacksburg, VA 24061, USA

Keywords: model reuse, study objectives; abstraction, model specification

INTRODUCTION

Reuse of existing models can significantly reduce development time and costs and improve the quality of newly developed simulations. However, reusing existing models is difficult in practice and is the focus of much research in the simulation community [1-7]. It is a central objective of HLA (High Level Architecture) developed by the U.S. Department of Defense. HLA is well documented at the Defense Modeling and Simulation Office website [8]. We believe that improving model reuse is a grand challenge in simulation. While the problems of model reuse are similar to those of code reuse in general, we suspect model reuse is more complex due to the centrality of abstraction in modeling. For example, because of different objectives, alternative models of the same physical system may use different abstractions that result in executions that differ in accuracy of outputs (high or low fidelity) or execution speeds.

Progress in model reuse will require significant development in several areas: 1) understanding what information is needed to support reuse and how it should be represented, 2) developing mechanisms, automated and manual, to collect and record this information, 3) understanding how to design for reuse, 4) developing analysis and search tools to locate appropriate existing components, 5) developing the ability to determine when model reuse is desirable.

We discuss some reasons for these difficulties and address some of what is needed to support reuse of existing components. Our goal is more to discuss several aspects of the reuse problem in the optimistic hope that articulation of a problem can make a contribution to its solution. However, some parts of the solution seem clear. For example, we argue that a key part of the solution must involve capturing the objectives, assumptions and constraints under which the original models were developed in a form that can be searched and analyzed.

Model Abstractions Based On Objectives

Much of what follows is based on the following assertion:

> All simulation is based on modeling to support some concrete objective (e.g. system performance is improved; my people are better trained to face this situation; I have a better understanding of what causes a system to do what it does, this is an enjoyable expenditure of my recreational time, etc.). Modeling is a process of abstraction, and abstraction involves simplification and omission of details. What simplifications are permissible and what details can be omitted depend on simulation objectives. No modeling is possible without consideration of some objectives. All valid reuse must take the objectives of existing components into account.

In addition, most simulations are developed within some set of constraints often with limitations on development time, budget, staff capabilities or performance capabilities. The end result often represents some compromise of conflicting objectives and constraints. Even when the same physical component is included in different simulations, differing objectives, assumptions, and constraints for the simulations can place differing, contradictory requirements on the behaviors associated with a particular object.

The traditional view of model development is that the models should be minimal; in the ideal case, they should be just sufficient to meet simulation objectives. This approach should generally produce simulations with less coding effort and better execution performance that are more easily validated. However, this focus on model minimality often makes reuse difficult. With faster and cheaper hardware, this emphasis on minimality can be inappropriate.

Model Reusability: Like DNA Transfer?

As with all software application domains, many parts of the source code that comprise an executable simulation are candidates for reuse. Some of the source code can be regarded as part of an infrastructure to support model execution and often has no counterpart in the system being simulated. These include, for example, data collection and statistical analysis routines, graphical generation tools for component representation, simulation utilities such as time management, random number generators, component communication and synchronization tools. Reuse of this code raises the same problems as software reuse in general; we choose not to address this further here. This is not to trivialize the problem, but its solution is an active research area in the software engineering community. Beyond code, much that is used to build a simulation is also potentially reusable in the development of new simulations, but our focus here is on model reuse.

We use the term *model* to refer to something that can create approximations of some important behaviors in a system of interest. We use the term *simulation* for an executable that includes a collection of executable models, the infrastructure necessary for their proper interaction, and any other code (for example, analysis or animation code) needed to meet simulation objectives. The models that exist in a simulation are sometimes called *submodels*. Each submodel generates some behaviors of interest in the complete simulation. We use the term *component* for both submodel code and other code that is part of the simulation.

Since some of these model behaviors may also be useful in other simulations, we are interested in what is required to install those behaviors in a new simulation. The easiest way to do this seems to be to reuse these submodels in the new simulation. For our purposes, we consider issues in the reuse of both the complete simulation model and the individual submodels.

The behaviors that are deemed important must be based on some stated or implied simulation objective. (Part of the art of modeling is identifying what behaviors are important for the objectives. Another traditional aspect of this art is creating the simplest model capable of producing those behaviors.)

Often a submodel corresponds to some aspects of an identifiable physical component in the system being simulated. Indeed, some modeling approaches suggest this as a basic approach to identification of submodels. However, often the submodel exists only as an abstraction with no identifiable correspondent in the simulated system; it is strictly a means to produce some necessary behaviors. For example, consider a system where component failures are of concern. A modeler might choose to generate these failures by creating, at least conceptually, a "demon" that roams the system causing components to fail.

DIFFERING NEEDS OF SIMULATION USERS

Several different communities now rely extensively on simulation. We expect this reliance to increase, particularly if hardware costs continue to fall and simulation development tools continue to improve. Consideration of the sometimes conflicting and differing perspectives of different simulation consumers raises important reuse issues:

??Simulations should evolve into “total immersion” virtual reality environments (to support military and industrial training along with the gaming community).

??Simulations should be built quickly (to support decision making by the military, for example).

??Simulations should be constructed as cheaply as possible to produce precise answers a particular question (to support acquisition or design decisions).

??Simulation should enhance a user's understanding of how a system works so that a decision maker can apply better judgment (in rendering business or design decisions).

??Simulations should be sensorly indistinguishable from reality with correctness less important (movie and gaming industries).

??Simulations should be able to produce invalid behavior, violating the laws of physics, for example, if such behavior enhances a story or sensual impact (movie and gaming industries).

??Simulations should accurately reproduce both anticipated and unanticipated behaviors of physical systems (to replace labs in the science research community and to support design decisions).

??Simulations must be made to run more efficiently (to support weather prediction, for example).

??Simulation should use simple familiar analogies, often graphical for it is the concepts rather than the details that matter (for tutorial purposes).

This list is not exhaustive but is meant to illustrate how different communities place different priorities on what is accomplished in a simulation study. And because of these dissimilar needs for execution speed, costs, visual/auditory representations, correctness, or accuracy, models used for simulations of the same physical object would likely be quite different in different communities.

With some exceptions, this list of dissimilar needs actually reflects differing priorities among these user communities. It implicitly assumes the necessity for trade-offs, mostly due to costs or speed limitations. If the executions were fast enough, development time quite rapid, or development costs sufficiently low, most of these compromises would be unnecessary.

However, anticipated dropping of hardware costs will not solve this problem. The authors have several decades of experience in the development of simulation models. One invariant across these decades is that as hardware performance has improved and costs have dropped, expectations of users and modelers have increased at least as quickly. Worse, perhaps: if past behavior is a reliable predictor of the future, the expected continuing drop in hardware costs will have a negative impact on reuse of today's models; in the near future, today's models will likely be perceived as insufficient for current needs.

Throughout the software development community, including simulation developers, we build systems for which a perceived need exists and that seem economically viable. Systems like HLA would likely not have been attempted in the past, not because they were undesirable but because the availability of instances of the necessary hardware infrastructure of fast processors and high-speed networks would have been perceived as too limited. Only recently have such networked systems become economically viable in sufficient numbers. Likewise, the success of graphically oriented simulation development products like Arena [9], Promodel [10], VSE [11] and many others require the existence of many inexpensive graphically capable high performance (at least by 1990's standards) PCs. These products make older text-based simulations seem largely obsolete. Certainly, newly developed simulations are much more likely to include animated representations than was the case in, say, 1990 or 1980.

In summary, perceived simulation users' needs are strongly influenced by the economics of hardware and development costs. This will work against reuse of current models. The trade-offs made today in different simulation communities will likely differ, but trade-off will still be necessary. This will continue to work against the reuse of models within and across different communities. This implies that reuse will continue to be based on the particular objectives, assumptions, and constraints associated with each existing model. Developing "one model of an object for all purposes" will still be unlikely. The ModSAF [12] community, for example, has addressed the reuse issue by using a "least common demoninator" approach: provide a collection of models in sufficient detail to meet all possible uses. The assumption is that the additional costs of running models with irrelevant details is less than the costs of developing new, hand-tailored versions. Changing hardware costs will still quickly make these models obsolete.

AUTOMATED SUPPORT OF REUSE

Many issues make a general automated solution to the reuse problem unlikely. Page and Opper [5] show that deciding whether an identified collection of submodels meet a stated set of objectives is NP-complete. Much earlier Overstreet and Nance pointed out that deciding if two submodels are functionally equivalent is undecidable [13]. Thus, any assistance in this area will be partial and require manual intervention or assistance.

Reuse has different meaning for different modelers. At one level, reuse might be limited to only recomposing existing models from a library; no modification of components will be performed. In other contexts, reuse can involve both reuse without modification, and include modifying an existing component if it is similar to what is needed and its use will speed development. Supporting completely automated composition of existing components into new models is a goal that can be achieved at best in very restricted domains with submodels designed with this in mind. Providing a set of tools to assist model/programmers in the construction of new simulation is a more realistic goal.

Key Issues

Several key issues must be addressed to support model reuse; some of these are discussed further below:

1. Determining how to locate potentially reusable components,
2. Recognizing objective incompatibilities among model components,
3. Recognizing assumption incompatibilities among model components,
4. Building components that enhance reuse,
5. Determining the level of granularity of each reusable component,
6. Capturing the objectives and constraints of each component,
7. Representing the objectives, assumptions and constraints,
8. Specifying the level of fidelity of each component,
9. Determining the modifiability of a reusable component,
10. Determining the interoperability of the reused components,
11. Determining if constraints (such as speed) will be satisfied with the selected objects,
12. If a new simulation is constructed entirely from verified/validated/accredited components, what can we say about the newly composed simulation?

Some of these problems can be addressed, in whole or in part, by capturing the objectives, assumptions, and constraints for each component. While it is not clear how these should be represented and what should be included, they can help address issues such as 1, 2, 3, and 8. Simple approaches such as key word searches are already in use. This can be facilitated by the development of standard taxonomies for describing individual components in particular problem domains. It is also possible to automatically extract features that might help identify key component attributes.

Determining the optimal granularity of submodels to be stored in a reuse library in order to facilitate broad reuse is complex. It includes the problem that has plagued simulation language designers from GPSS forward: if the components are at too low a level, their reuse requires much the same effort as coding from scratch. If the components are high level aggregates, then their reuse may be limited; new simulations may require the creation of new versions of existing submodels. Many current simulation languages address this problem in part by providing a rich collection of easily parameterized components. The language really consists of a collection of reusable components and the infrastructure to support their execution and interaction. Programming new simulations is accomplished by selecting, connecting, and parameterizing components from the language-provided collection.

Capturing necessary assumptions, objectives and constraints can be difficult. Often assumptions that determine what are important and unimportant behaviors of a physical system (and then influence what is included and excluded in the model of that system) are implicitly included. In our experience, the modeler may be unaware that such assumptions have been made until it is necessary to explain what is needed to a new programmer or modeler. After one has worked in a simulation domain for a while, these assumptions can become internalized and efficiencies are gained since they do not need to be stated to other members of the same domain.

One approach for creating a reuse library is to build a set of components from scratch. This is no less a challenge than the task of creating a new simulation language and is a daunting task if the goal is provide components sufficient for the construction of any simulation. The approach seems more tractable if the domain of interest is restricted.

ModSAF [12] and OneSAF [14], derived from ModSAF, have a long development history by DoD and are examples of building a collection of components specifically for reuse. For both systems, a collection of components (along with a supporting infrastructure for execution and interaction of components) was developed. Each component is developed based on a common subset of objectives so that they can be composed into new simulations at will. The compatible objectives should ensure that all needed interactions among these components can occur and that interactions will be correct. After starting the simulation, a user employs a menu to select the components to be included and to assign tasks to them; the system can also read this from configuration files. While performance, flexibility, and extensibility have been issues, and the approach is expensive, the system has been used to conduct many simulations involving different combinations of components and objectives. However, one can argue whether these systems are examples of component reuse or examples of highly configurable systems. Taking components out of ModSAF and reusing them in a completely new simulation would probably be no different from other forms of software reuse. Issues of incompatible assumptions, objectives, and constraints must still be addressed.

Cost Issues

Given the diverse needs of different simulation communities, we should avoid the "one solution fits all" approach to model reuse. For many small simulations, much of the information about assumptions, objectives, and constraints associated with existing components may be carried in the minds of modelers and programmers. Having a written record is beneficial but the costs of recording them needs to be balanced against its benefits. Code reading and manual searches may be more cost effective in many cases.

For larger systems, this information needs to be captured in a more formal manner. For modestly sized systems, it may be sufficient if captured as human readable documentation. For larger systems, it may need to be captured in machine analyzable form. But regardless of simulation size, these efforts should be influenced by the costs associated with the unrecognized use of invalid models.

Carefully documented constraints, assumptions, and objectives have benefits beyond support of reuse. For example, such description is necessary for verification, validation and accreditation (VV&A) [15]. But for some simulations, VV&A is done informally; based on subjective criteria; if the behaviors look good enough, they are good enough. For other uses of simulation, such as exploring new concepts or gaining insights, no formal validation may be needed; the understanding gained through building a functioning model may be the primary objective. Carefully capturing objectives, constraints, and assumptions might not be worthwhile.

Given the diversity of objectives in building a simulation, building a single model of a system component sufficiently generic to satisfy all simulation needs seems impossible. Even if possible, this approach will not be economically viable in many cases. Universal Interoperability (the term used in HLA for model reuse) is a stated HLA goal, but it is often inappropriate, even for models built for a single organization.

Benefits of Abstraction

An interesting question in the construction of reusable components is the benefit and importance of abstraction. The simplifications that usually result from abstraction can result in faster simulation execution, quicker code development, easier modification and revision, and easier understanding of what is being simulated.

From one perspective, creating a high level abstraction for a variety of components can facilitate reuse, since multiple instances of the same abstraction, varying only in cosmetic details, may appear often. However, this approach may often require tailoring of the abstraction for most reuses. It is not clear if we would reduce costs and improve quality by providing implementations of a variety of commonly used abstractions, e.g. M/M/1 queues, while allowing different “skins” or wrappers to provide different visual representations. Identifying a collection of often occurring abstractions would take some effort. In addition, designing them so that they are easily tailored requires additional understanding. If most reuse requires developing different realistic appearances, little may be gained in the approach.

SUMMARY

Model reuse is an important goal since it can reduce development time, decrease costs, and improve the quality of simulation models. However, understanding how to design models to facilitate reuse is a challenge. How to decide if candidate existing models will satisfy a given set of objectives is difficult; automated solutions are computationally intractable. The diversity of objectives for different simulation uses makes creation of models that can satisfy all simulation needs infeasible. The changing economics resulting from cheaper faster hardware will work against future reuse of existing models. Likewise, mandating a single approach intended to facilitate reuse, even within a single organization, can be economically inappropriate due to the diversity of uses of simulations.

Significant model reuse has been successfully demonstrated in several projects, but in limited domains and is either based on models developed from a common set of objectives or has required significant software development to enable it. At present, the costs of these approaches restrict their use to only a few problem domains.

A key to reuse is the capturing of objectives, assumptions, and constraints associated with each component. Given the constantly changing expectations of simulation users, the economics of reuse may be as much in the form of identifying components for modification to meet new objectives than in locating components that admit reuse without modification.

Reference List

[1] S. Ferenci, K. Perumalla, and R. Fujimoto, "An Approach for Federating Parallel Simulators," *Proceedings of the 14th Workshop on Parallel and Distributed Simulation (PADS 2000)*, pp. 63-70, May 28-31, 2000

[2] J. Aronson and Prasanta Bose, "A Model-Based Approach to Simulation Composition, *Proceeding of the Fifth Symposium on Software Reusability*, pp. 73-82, 1999.

[3] J. Dahmann, R. Fujimoto, and R. Weatherly, "The DoD High Level Architecture: An Update," *Proceedings of the 1998 Winter Simulation Conference*, pp. 797-804. 1998.

[4] G. Mackulak and F. Lawrence, "Effective Simulation Model Reuse; A Case Study for AMHS Modeling," *Proceedings 1998 Winter Simulation Conference*, pp. 979-984, Dec. 1998.

[5] E. Page and J. Opper, "Observations on the Complexity of Composable Simulation," *Proceedings of the 1999 Winter Simulation Conference*, pp 553-560, 1999.

[6] S. Kasputis and H. Ng, "Composable Simulations," *Proceedings of the 2000 Winter Simulation Conference*, pp. 1577-1584, Dec. 2000.

[7] O. Balci, A. Bertelrud, C. Esterbrook, R. Nance, "Developing a Library of Reusable Model Components by Using the Visual Simulation Environment," *Proceedings of the 1997 Summer Computer Simulation Conference*, pp. 253-258, July 1997.

[8] Defense Modeling and Simulation Office, http: //hla.dmso.mil.

[9] http://www.arenasimulation.com.

[10] http://www.promodel.com.

[11] http://www.orcacomputing.com.

[12] http://modsaf.org

[13] C. Overstreet and R. Nance, "World View Based Discrete Event Model Simplification," in *Modelling and Simulation Methodology in the Artificial Intelligence Era,"* M. Elzas, T. Ören, B. Zeigler (Eds.), North-Holland, pp. 165-179, 1986.

[14] http://onesaf.org.

[15] O. Balci and W. Ormsby, "Well-Defined Intended Uses: An Explicit Requirement for Accreditation of Modeling and Simulation Applications," *Proceedings of the 2000 Winter Simulation Conference*, pp. 849-854, Dec. 2000.

C. Michael Overstreet is an Associate Professor of Computer Science at Old Domionion University. A member of ACM and IEEE/CS, he is a former chair of SIGSIM, and has authored or co-authored over 80 refereed journal and conference articles. He received a B.S. from the University of Tennessee, an M.S. from Idaho State University and an M.S. and Ph.D. from Virginia Tech. He has held visiting appointments at the Kyushu Institute of Technology in Iizuka, Japan, and at the Fachhochschule fur Technik und Wirtschaft in Berlin, Germany. His current research interests include model specification and analysis, static code analysis and support of interactive distance instruction. Dr. Overstreet's home page is www.cs.odu.edu/~cmo. He can be reached by e-mail at cmo@cs.odue.edu.

Richard E. Nance is the RADM John Adolphus Dahlgren Professor of Computer Science and the Director of the Systems Research Center at Virginia Tech. Dr. Nance is also Chairman of the Board of Orca Computing. He has held research appointments at the Naval Surface Weapons Center and at the Imperial College of Science and Technology (UK). He has held several editorial positions and is the founding Editor-in-Chief of the *ACM Transactions on Modeling and Computer Simulation.* He served as Program chair for the 1990 Winter Simulation Conference. Dr. Nance received a Distinguished Service Award from the TIMS College on Simulation in 1987. In 1995 he was honored by an award for "distinguished Service to SIGSIM and the Simulation Community" by the ACM Special Interest Group on Simulation. He was named an ACM Fellow in 1996. His e-mail and web addresses are nance@vt.edu and www.cs.vt.edu/info/people/vitae/Nance.html.

Osman Balci is Professor of Computer Science at Virginia Tech and President of Orca Computing, Inc. He received his Ph.D. from Syracuse University in 1981. Dr. Balci is Editor-in-Chief of two international journals; *Annals of Software Engineering* and *World Wide Web*; Verification, Validation, and Accreditation Area Editor of the *ACM Transactions on Modeling and Computer Simulation*, and Modeling and Simulation Category Editor of the *ACM Computing Reviews.* He is a member of the Winter Simulation Conference Board of Directors representing the Society for Computer Simulation. DoD has funded most of Dr. Balci's research since 1983. Recently he has provided technical services for the National Missile Defense program in the areas of system design. His current research interests center on VV&A, IV&V, certification, quality assurance and credibility assessment of M&S applications, software systems, and complex software/hardware/humanware system designs. His e-mail and web addresses are balci@vt.edu and manta.cs.vt.edu/balci.