Proceedings of the 1998 Winter Simulation Conference
D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, eds.

THE FUTURE OF JAVA-BASED SIMULATION

Richard A. Kilgore
Kevin J. Healy

ThreadTec, Inc
P. O. Box 7
Chesterfield, MO 63017, U.S. A

George. B. Kleindorfer

Smeal College of Business Administration
Penn State University
University Park, PA 16802, U.S.A.

ABSTRACT

Java-based simulation presents a unique opportunity for revolutionary changes in the process of developing simulation models and in the mission of the simulation software firms that provide tools to support the model development process. Java enables a new vision of a simulation industry populated by application-specific simulation specialists who generate compatible and reusable simulation components. These object-oriented components can be developed using inexpensive, professional-quality Java development environments and executed using Internet browser software. This discussion is an overview of the features and future benefits of Java-based simulation. It is targeted at experienced simulation practitioners who understand the limitations of existing tools and the need for object-oriented, standardized and reusable modeling software.

1 INTRODUCTION

The Java programming language has a number of features that have the potential to dramatically change the process used to build computer simulation models. Previous discussions of these features have focused on explanations of the technical differences between Java and other object-oriented approaches to simulation (Buss and Stork 1996; Healy and Kilgore 1997; Kilgore et al. 1998). This discussion is an overview of these features in an introductory format for those prac-titioners that understand simulation but do not yet comprehend the far-reaching opportunities of object-oriented modeling within industry-standard Java development environments. If realized, the benefits for modelers include:

- Java will become a common foundation for all sim-ulation tools because it is the only object-oriented programming environment that effectively supports standardized components.
- Java will foster execution speed breakthroughs through convenient and robust support for distri-buted processing of simulation experiments on multiple processors.
- Java will improve the quality of simulation models as the development of application-specific software components redirects the emphasis of simulation software firms toward modeling and away from modeling development environments.
- Java will expose the benefit of computer sim-ulation to a larger audience of problem-solvers, decision-makers and trainers since models can be distributed and executed over the internet using standard browser software on any operating system and hardware platform.
- Java will accelerate simulation education because students already familiar with object-oriented design, Java syntax and Java development environ-ments will no longer require instruction in specific simulation tools.

While some of these opportunities may take decades to fully realize, many of these benefits have already been experienced in early implementations of the Silk simulation tool (Healy and Kilgore, 1998). It is not within the scope of this paper to describe Java (Grand 1997) or the specifics of the Silk imple-mentation of Java-based simulation. In fact, a primary objective here is to illustrate that Java encourages a vendor-neutral simulation environment where Silk may be just one of a number of commercially available collections of Java-compatible simulation components. No programming language or simulation tool seems to have more than a 30-year life span, and it is un-reasonable to expect that Silk and Java will not eventually be superseded by even more superior tools. But the introduction of "open" programming tools like Java and "open" Java-based simulation tools like Silk will be the beginning of the end to the limitations im-posed

by the lack of compatibility that exists in current simulation software technology.

2 JAVA WILL BE THE FOUNDATION OF ALL SIMULATION TOOLS

Just as industrial engineers employ simulation to study and improve a process, industrial engineering should be applied to the process of simulation. One of the peculiar aspects of the process of simulation is that the "standardization of components" which led to the industrial revolution in the early 20th century seems to be lacking. Model building remains the product of craftsmen who work an entire piece from start to finish using specialized tools and techniques which are rarely passed down to their apprentices. Java has the oppor-tunity to be the catalyst for a simulation industrial revolution because it is the first programming language that can foster the growth of platform-neutral and vendor-neutral standardized simulation components.

There are a number of features of Java which enable the creation of standardized simulation com-ponents and understanding of these features will be crucial to the emergence of accepted standards and design patterns. The most significant of these features is Java's simple and straightforward approach to multi-threaded programming (Oaks and Wong, 1997).

The importance of multithreaded programming for simulation is that it enables the creation of an entity-thread that has independent control of its behavior within the simulation. Unfortunately, multithreaded programming is a new programming concept to most simulation practitioners (who are too busy learning the latest ver-sion of their vendor-specific software), and the sig-nificance of the entity-thread concept is not readily understood.

Most software programs run in a single thread of execution. In such an environment, there is a single intelligent entity that is controlling the execution of each program instruction. In multithreaded execution, there are multiple, independent, intelligent entities that share control of the execution of program instructions. Similar to multitasking operating systems, in which multiple programs are allowed to execute simul-taneously in separate address spaces, multithreaded programs are executing simultaneously within the same program. Of course, on a single processor computer, only one thread is actually active at any one time and other threads are suspended until specified conditions are met which allow them to complete their individual tasks.

It should be clear that the concept of multi-threaded programming better corresponds to the operation of the real world system being simulated. Most systems are better described as a collection of independent, intelligent entities rather than a single global entity agent that must decide when and where to allocate program control. Programming a simulation within a single-threaded environment is similar to an orchestra in which the conductor is responsible for directing and playing each note. Not only does it place a great deal of responsibility on the part of the conductor, it does not allow the assembly of an orchestra using independent, easily changeable musicians and instruments.

The practical significance of multithreaded pro-gramming and entity-threads is that there is no longer the need for a simulation language to serve as the agent coordinating the flow of control within the simulation. A Java-based simulation can generate an entity-thread object that corresponds to an intelligent entity in the system being simulated. Rather than using a programming language to construct a simulation lan-guage to emulate the completion of the entity tasks, a Java-based simulation results from the *direct* execution of entity-thread process methods. Java is not just an underlying programming language for simulation. Java is the simulation language.

Just as a traditional programming language pro-vides fundamental behaviors such as,

do if start read write

a Java-based simulation adds simulation-specific entity behaviors such as,

create queue seize delay release

which are methods executed *directly* on the entity-thread. An entity-thread executes the *queue* method just as it executes the *read* statement. There is no compilation of simulation statements into programming instructions. And there is no black-box, simulation-language-specific representation of an entity executing simulated process steps.

Most importantly, if the simulation engine sup-porting entity-threads is properly implemented, anyone with knowledge of Java and the simulation API can extend the simulation tool through the modification of existing methods or the addition of new methods. Since the simulation engine is itself a component, it is even possible to replace it if another variety was more suited to a particular application.

The consequence for the simulation industry is that Java eliminates the need for simulation vendors to distinguish themselves through the creation of incom-patible software and development environments. All simulation tools could share a common set of standards and simulation program development would consist of the assembly of reusable and compatible simulation components based on these standards.

3 JAVA WILL FOSTER EXECUTION SPEED BREAKTHROUGHS

A favorable by-product of multithreaded pro-gramming will be the execution speed improvement made possible through the distributed execution of Java-based simulation on multiple processors. Returning to the industrial engineering of the process of discrete-event process simulation, consider that each replication of the simulation is done sequentially on the single processor. Regardless of the speed of the in-dividual replication, significant execution speed break-throughs exist only through the simultaneous execution of different replications (with different random number seeds) on different processors. While Java presently does not directly support multiple processors, the ability to separate simulation programs into entity-threads is a convenient first step along that path until the multiprocessor API is released. Java's internal support for distributed network programming provides the remainder of the necessary infrastructure to achieve distributed simulation experiment processing without making the simulation dependent on the operating system or hardware platform.

While distributed processing of simulation experiments is a likely to be near-term application of Java-based simulation, a more interesting long-term opportunity is the allocation of a separate processor to each entity-thread. There is obvious benefit simply from the execution speed improvements due to parallel processing of entity-thread processes within the simulation run (Ferscha and Richter, 1997). But of greater interest is the elegant parallel between entity-thread-processor based computer simulation and the entities in the real systems being simulated. Just as entity-threads better represent the behavior of independent, intelligent agents in a single-processor system, entity-thread-processors provides an even better correspondence between behavior in the real system outside of the computer and behavior in the simulated system within the computer. Presently, we must emulate the separate processors in software using vendor-specific prioritization schemes that allocate the single processor to competing entity-threads. The natural evolution of this approach would be to eliminate the overhead of entity-thread management through the use of multiple processors.

4 JAVA WILL IMPROVE THE QUALITY OF SIMULATION MODELS

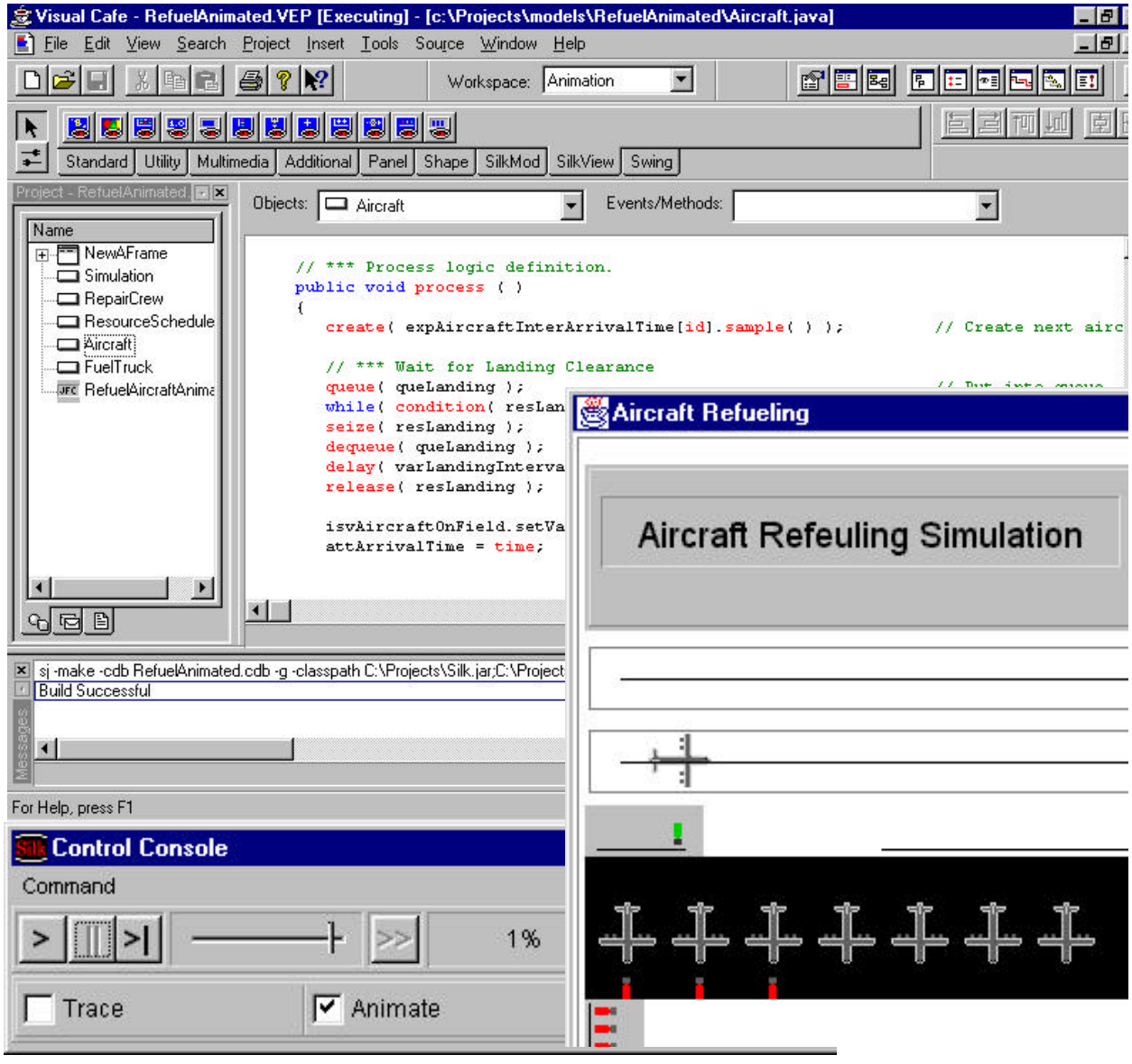
The industrial engineering study of the process of simulation now turns towards the impact of Java-based simulation on the quality of the simulation model product. This study focuses less on the activities of the simulation developer in the field and more on the activities of the simulation software vendor. An analysis of these programming activities at present simulation software vendors would lead to the conclusion that more effort is devoted on the interface to the simulation software than is spent developing usable and reusable applications of the simulation software. Java-based simulation would reverse this situation by promoting simulation model development using off-the-shelf Java development environments.

Presently available Integrated Development Environments, such as Symantec Visual Café for Java shown in Figure 1, eliminate the need for a simulation software vendor to maintain expertise in programming areas outside of the domain of simulation. The tools for editing, project management, debugging, visual modeling and documentation available in these en-vironments are far superior to any similar functions in presently available simulation software. And because these are off-the-shelf development environments used throughout the world, there is already a population of trained users familiar with the operation of these tools. Since the developers of Java IDEs have a much larger user population than just simulation users, a larger amount of resources will be invested in the future improvement of these tools.

Prior to Java-based simulation, a simulation software vendor must allocate programming resources to four activities:

- Internal simulation engine and simulation language
- Integrated development environment and GUI
- Animation and other input/output options
- Application-specific simulation components

In a Java-based simulation environment, it is no longer necessary to devote as much programming resource to the first three activities. These functions will be adequately available and better supported using commercially available, third-party software built for Java applications. This frees valuable simulation development resources for the creation and distribution of higher quality, application-specific simulation tools.



The bigger quality impact of this new role for simulation software vendors will be derived from the increased economic incentive to create more reusable simulation components. In the present simulation marketplace, simulation vendors profit from the eventual incompatibility of previous versions (or sections of previous models) with future simulations. Given that Java-based simulation components could be easily distributed, modified and resold throughout the world over the internet presents an entirely different "model" to the primary business of a simulation soft-ware vendor. Simulation software consultants can now afford to specialize in specific applications of simulation and become direct software product vendors rather than simply services vendors.

There are many other benefits to simulation software developers from the use of Java. The need to maintain different versions of the software is eliminated since Java offers cross-platform compatibility. Vendors can release bug-fixes immediately through the distribution of a single class file rather than postpone fixes and improvements to the next release. Similarly, since the Java simulation code is conveniently packaged into modular classes, it is very convenient to develop, distribute and maintain different versions for different industries and different customers.

5 JAVA WILL EXPOSE THE BENEFIT OF SIMULATION TO A LARGER AUDIENCE

Suppose that 10 years ago you were told that it would be possible to save an animated simulation model to your local hard drive and have that model become instantly visible and usable by millions of people around the world. And then suppose you were told that the browser software to enable this capability was available at no cost.

The belief that Java will emerge as a standard for simulation programming is further advanced by the close relationship between Java and internet-based computing. More and more, communication between people and businesses will occur through network connections. Telecommuting, virtual corporations and the internationalization of business will continue the trend toward the transmission of ideas through electronic media. What better mechanism for communicating information about dynamic systems than through the online distribution of dynamic computer simulations that depict these systems in operation.

For example, consider Bob in Boston who wants to use simulation to describe a new material handling option to Chris in California. He must either fly with his laptop to make a presentation in person, or have Chris purchase and install the required simulation software on compatible hardware. Using Java-based simulation software, Chris can use her browser software to link to Bob's site where the simulation is located and view the system online while she discusses improvements with Bob over the phone.

This ability to execute within the Internet browser is the feature of Java that will enable Java-based simulation to realize its long sought potential as an online demonstration and training tool. Rather than waste time and resources at onsite demonstrations and industrial shows, equipment vendors can show the simulated operation of their products to clients throughout the world and throughout the year. Plant personnel can have immediate access to a simulation of the projected daily activity within the facility. Educators and trainers can conveniently and inexpensively make textbooks come alive with online simulation models of industrial, physical and biological systems.

And as more people become exposed to the benefits of computer simulation, more support for the technology will emerge, as the quality of the online simulation becomes a comparative advantage between competing firms.

6 JAVA WILL ACCELERATE SIMULATION EDUCATION

Java-based simulation will establish simulation programming within the mainstream of computer science education. Students and professors will no longer need to learn simulation-vendor-specific syntax or learn to use a simulation-vendor-specific develop-ment environment. Familiar Java development environments that students used in prerequisite com-puter science classes will become simulation platforms. More importantly, the principles of programming and object-oriented design learned in prior study will transfer directly into their work with Java-based simulation.

Another important feature of Java for education is that it eliminates the difficult trade-off between easy-to-use simulators and powerful and flexible simulation languages. Those students with sufficient pro-gramming skills can combine simulation and Java programming without limitation. Other students with less programming ability will have a rich library of reusable, simulation components available on the Internet as a starting point for their projects. Even those students and professors whose interest does not extend past visual modeling can make use of JavaBeans (Kilgore 1998) visual component models.

The leadership of simulation educators and researchers will be a critical factor in the transition to Java-based simulation as it will likely influence the speed at which consensus is reached for the emergence of standards for object-oriented simulation components. Simulation is a relatively immature industry populated by numerous small firms, so it is unlikely that the movement towards standardization will be the product of an industry association. It is much more likely that Java-based simulation standardization would be the outcome of government programs that can provide the legal and economic incentives, or academic programs that will produce the next generation of simulation analysts eager to take advantage of new technologies.

Finally, simulation education and research will also be accelerated through the convenient merger of other related Java-based applications with Java-based simulation. Students and researchers will be en-couraged by the ease that other Java-based software and Java-compatible databases can be merged into the creation of comprehensive modeling systems (Reese 1997). Presently, the task of creating interfaces to the current simulation-vendor-specific software requires difficult, custom programming that has little oppor-tunity for reuse. The integration of simulation with other planning, analysis, scheduling and control applications can be accomplished through the creation of modular, object-oriented Java class libraries that can be easily added and modified to work with existing Java-based simulation classes.

7 SUMMARY

How many models of systems similar to those already simulated are started each day around the world? How many of those previously completed models are discarded? How many developers can use portions of previous models created 10 years ago using the current version of the

software being supplied by their software vendor? How many developers can organize large models into modular components and package those components into independent files that contain all of the data and methods needed to describe that portion of the model? And even if this component can be created, how many developers can share these components with others that have not chosen their brand of simulation software? Eventually, the wasted investment in obsolete discrete-event, process simulation software and incompatible models will lead to the demand for new, more efficient and economical approaches.

This paper presented the argument that the Java programming language presents an opportunity to serve as the foundation for the next generation of computer simulation development in the 21st century. Due to limitations in existing programming languages and development environments, simulation in the 20th century has been the product of a single developer managing the assembly of a single software element that models the progress of a single entity thread on a single processor. Because of the Java programming language, computer simulation in the 21st century has the opportunity to be the product of multiple developers managing the assembly of multiple software elements that model the progress of multiple entity threads on multiple processors.

REFERENCES

Buss, A. H. and K. A. Stork. 1996. Discrete-Event Simulation on the World Wide Web Using Java, *Proceedings of the 1996 Winter Simulation Conference*, IEEE, Piscataway, NJ.

Grand, M. 1997. *Java Language Reference, 2nd Ed*. O'Reilly & Associates, Inc., Sebastopol, CA.

Healy, K. J. and R. A. Kilgore. 1997. Silk: A Java-Based Process Simulation Language. *Proceedings of the 1997 Winter Simulation Conference*, IEEE, Piscataway, NJ.

Healy, K. J. and R. A. Kilgore. 1998. Introduction to Silk and Java-Based Simulation. *Proceedings of the 1998 Winter Simulation Conference*, IEEE, Piscataway, NJ.

Kilgore, R. A. 1998. *Introduction to Visual Modeling with Silk and JavaBeans*. ThreadTec, Inc., St. Louis, MO. White paper available at www.threadtec.com.

Kilgore, R., K. Healy and G. B. Kleindorfer. 1998. Silk: Usable and Reusable, Java-Based, Object Oriented Simulation. *Proceedings of the 12th European Simulation Multiconference.* Society for Computer Simulation International, Ghent, Belgium.

Oaks, S. and H. Wong. 1997. *Java Threads*. O'Reilly & Associates, Inc., Sebastopol, CA.

Reese, G. 1997. *Database Programming with JDBC and Java*. O'Reilly & Associates, Inc., Sebastopol, CA.

Rothenberg, J. 1986. Object-Oriented Simulation: Where Do We Go from Here? *Proceedings of the 1988 Winter Simulation Conference*. IEEE, Piscataway, NJ.

AUTHOR BIOGRAPHIES

RICHARD A. KILGORE is a partner in ThreadTec specializing in the design and development of application-specific simulation components using Silk. He received his Ph.D. in Management Science from Penn State University. He has over 15 years of experience as a modeling consultant to Fortune 500 firms in a variety of industries. Formerly, he was a capacity-planning analyst with Ford Motor Co. and Vice-President of Products for Systems Modeling Corp.

KEVIN J. HEALY is the author of the Java-based Silk simulation language and a partner in ThreadTec. He received his Ph.D. in Operations Research from Cornell University. He was formerly a member of the Indus-trial Engineering faculty at Purdue University and was Vice-President of Development for Systems Modeling Corp. He was an Associate Editor for the Proceedings of the 1997 Winter Simulation Conference.

GEORGE B. KLEINDORFER is an emeritus professor and former chair in the department of Management Science and Information Systems at Penn State. His research and teaching have centered on simulation, control theory and the philosophy of science. He has taught simulation in business and engineering for three decades. At Penn State, he was named the Alumni Faculty Fellow in 1991, the most prestigious recognition of teaching accomplishment at that university.