Web-Based Simulation: Revolution or Evolution?

ERNEST H. PAGE
The MITRE Corporation
ARNOLD BUSS
Naval Postgraduate School
PAUL A. FISHWICK
University of Florida
KEVIN J. HEALY
ThreadTec, Inc.
RICHARD E. NANCE
Virginia Tech
and
RAY J. PAUL
Brunel University

The nature of the emerging field of web-based simulation is examined in terms of its relationship to the fundamental aspects of simulation research and practice. The presentation, assuming a form of debate, is based on a panel session held at the first International Conference on Web-Based Modeling and Simulation, which was sponsored by the Society for Computer Simulation during 11–14 January 1998 in San Diego, California. While no clear "winner" is evident in this debate, the issues raised here certainly merit ongoing attention and contemplation.

Categories and Subject Descriptors: I.6.5 [**Simulation and Modeling**]: Model Development—*Modeling methodologies*; I.6.8 [**Simulation and Modeling**]: Types of Simulation—*Distributed*

General Terms: Design

Additional Key Words and Phrases: Digital objects, distributed modeling, Java

Authors' addresses: E. H. Page, The MITRE Corporation, 1820 Dolley Madison Blvd., McLean, VA 22102; A. Buss, Operations Research Department, Naval Postgraduate School, Monterey, CA 22102; P. A. Fishwick, Department of Computer and Information Science Engineering, University of Florida, Gainesville, FL 22102; K. J. Healy, ThreadTec, Inc., P.O. Box 7, Chesterfield, MO 63017; R. E. Nance, Systems Research Center and Department of Computer Science, Virginia Tech, Blacksburg, VA 24061; R. J. Paul, Centre for Applied Simulation Modeling, Brunel University, Uxbridge, Middlesex UB8 3PH, UK.

1. PREFACE

The emergence of the World Wide Web (WWW) has enabled—or caused—many people from across a wide range of disciplines to reevaluate their inherent business practices: the approaches, techniques and philosophies they apply in their day-to-day activities. The disciplines concerned with computer simulation are no exception to this phenomenon; the concept of *web-based simulation* has been introduced and is currently the subject of much interest to both simulation researchers and simulation practitioners. As an area of "scholarly" endeavor, web-based simulation made its debut as a 3-paper session at the 1996 Winter Simulation Conference (WSC) and was, by far, the most well-attended session within the modeling methodology track of that conference. This success was repeated at WSC 1997, and in January 1998 the first conference dedicated to the topic of web-based simulation was held as part of the annual Society of Computer Simulation (SCS) Western Multiconference [Fishwick et al. 1998].

This paper stems from a panel convened for WEBSIM '98. The charter for the panel was to examine the *fundamental* nature of web-based simulation and explore its relationship to the body of theory and practice in simulation modeling methodology that has evolved over the past forty years [Page et al. 1998]. One goal for the panel was to distill the essential and differentiating aspects of web-based simulation, if any, from amongst the mountains of hype that tend to surround the WWW. The central question was this: does web-based simulation represent a revolutionary change or an evolutionary change? We posed the question because the nature of this change would seem to dictate the proper direction and focus for web-based simulation research and practice.

The panel composition was structured in an effort not only to portray all sides of the issues being addressed but also, hopefully, to engender controversy and stimulate participation of the audience. Arnold Buss, Paul Fishwick, and Kevin Healy are active in web-based simulation research and development. Dick Nance and Ray Paul represent the "traditional" simulation modeling methodology community. Kevin Healy presents a view from the commercial world, the rest of the panel hails from academe. Arnold Bass, Dick Nance, Paul Fishwick, and Kevin Healy provide a U.S. perspective. Ray Paul serves as international representative.

The remainder of the paper is organized as follows. Section 2 establishes the framework for our debate. The panelists' responses are captured in Section 3. Although difficult to portray in linear text, we attempt to capture the essence of the dialogue that occurred during the session through focus on a few key points of dispute in Section 4. A brief concluding summary is given in Section 5.

2. WHAT ARE THE MODELING METHODOLOGICAL IMPACTS OF WEB-BASED SIMULATION?

Simulation modeling methodology deals with the creation and manipulation of models over the lifetime of their use. Motivated by the recognition

that the manner in which a simulation model is *conceived, developed* and *used* can have a significant impact on the ability of the model to achieve its objectives, modeling methodology has been an active research area since the inception of digital computer simulation.

Over the past forty years the practice of simulation model creation has evolved from coding in general-purpose languages, to model development in special-purpose simulation languages, to model design using higher-level simulation model specification languages and formalisms, to comprehensive theories of simulation modeling and holistic environment support for the modeling task. Thematic in much of the modeling methodological work to date has been the recognition of Dijkstra's principle of the "separation of concerns," which argues for the separateness of specification and implementation [Dijkstra 1976]. In many cases, this philosophy has been tempered by the pragmatic observations of Swartout and Balzer [1982], who observe that separation is a worthy goal but not achievable in totality since any specification, S, may be viewed as an implementation of some higher-order specification, S'.

At first glance, it would seem that technology is a kind of "raw material," providing the framework for implementation issues. As such, we should therefore view technology as *separate* from the fundamental aspects of a given problem, e.g., modeling methodology. On the other hand, one must consider the role of technology as an enabler. Specifically, we recognize that technological advance can permit *revolutionary* change in the way we approach problems. The advent of the assembly line in manufacturing is an often-cited example of this phenomenon. Is the WWW a "revolutionizing" technology? According to Erkes et al. [1996], "Our initial experiments at putting engineering, design and manufacturing services on the Web are so successful that we believe we should rethink the traditional approaches and tools for coordinating large, distributed teams." With respect to simulation, a similar revolution seems plausible. Web technology has the potential to significantly alter the ways in which simulation models are developed (collaboratively, by composition), documented (dynamically, using multimedia), analyzed (through open, widespread investigation) and executed (using *massive* distribution).

Revolution or evolution? Is the web a technological elixir, demanding that we radically alter our modeling philosophies and approaches? Or is web-based execution merely another implementation detail that can, and should, be abstracted from the model development process?

3. RESPONSES

The following sections contain responses from the panel members regarding the central question.

3.1 Web-Based Simulation Modeling (Arnold Buss)

The explosion of computer networks has created an environment for computer modeling in general, and simulation modeling in particular, that

is revolutionary. In order to properly exploit these developments the nature of modeling must change. Simulation models have been traditionally monolithic in design. The advent of Object-Oriented Programming has resulted in more elegantly designed monoliths. Simulation models for both industrial and military applications have been mostly designed for models running on a single machine. For such models the network offers little. Using the full power of the network offers potentially substantial benefits to modeling and simulation, but only if models are designed differently.

Use of up-to-date data by dynamically interacting with databases across the network speeds up the modeling and decision-making cycle by an order of magnitude. The integration of computer models running with systems has great potential for military analysis and training.

In a nutshell, web-based simulation models must accommodate:

—applications that expect to receive data across the network from a database that will be dynamically determined,

—applications that will expect to receive new classes and data unforeseen at the time the model was started, and

—applications using components that are loosely bound, rather than tightly coupled.

The Java programming language, together with the related cluster of Java Technologies, have substantially extended the capabilities of program-level tasks. Java classes can open sockets across a network, perform database queries, and encrypt data streams for secure transmission. New classes may be dynamically incorporated as the program is running, thus enabling dynamic extensibility. Objects on one computer may be serialized and sent to another, where they are immediately incorporated into that computer's model. Objects on another computer may be invoked through Remote Method Invocation.

The capabilities of programming languages have outstripped our knowledge of how best to write programs exploiting these capabilities. Software design principles for procedural and even object-oriented programs are well known. It is not yet known how software should be designed using these tools. It is also not clear how best to exploit the tremendous possibilities offered by the network.

3.2 Distributed Modeling Using the Web as an Infrastructure (Paul A. Fishwick)

One of the most critical problems in the field of computer simulation today is the lack of published models and physical objects within a medium—such as the World Wide Web—allowing such distribution. The web represents the future of information sharing and exchange, and yet it is used primarily for the publication of documents since the web adopts a "document/desktop metaphor" for knowledge. In the near future, we envision an "object metaphor" where a document is one type of object. A web predicated

on digital objects is much more flexible and requires a knowledge in how to model physical phenomena at many different scales in space-time.

If a scientist or engineer (i.e., model author) works on a model, places the model inside objects and constructs a working simulation, this work occurs most often within a vacuum. Consider a scenario involving an internal combustion engine in an automobile, where the engine is the physical object to be simulated. The model author's task is to simulate the engine given that a new engineering method, involving a change in fuel injection for example, is to be tested. By testing the digital engine and fuel injection system using simulation, the author can determine the potential shortfalls and benefits of the new technique. This task is a worthwhile one for simulation, and simulation as a field has demonstrated its utility for objects such as engines.

Let's analyze the problems inherent in this example. There is no particular location that will help the author to create the geometry of the engine and its dynamics. Moreover, if the model author seeks reusable components on the web, who is to ensure the quality or accuracy of these components? It may be that other employees of the company have made similar engine models in the past, and that these models may be partially reused. If this is the case, the model author is fortunate, but even if such a company-internal model exists, it may not be represented in "model form." There may be other model authors who have already constructed pieces that our model author could use, but if there is no reuse and no standard mechanism for publishing the model or engine object, then this is all for naught. The model author may also be concerned with creating a fast simulation. While algorithms for speeding simulations are important, by solving the reusability problem, we also partially solve the speed problem, since published quality models of engines will battle in the marketplace for digital parts, and the best engine models and testing environments—involving very fast and efficient simulation algorithms—will win out in the end. Therefore, the problem of reuse of engine objects and components lies at the heart of the simulationist's dilemma. Fast, efficient, and quality models could be available at some point in the future, but today there are no infrastructure or agreed-upon standards for true digital object engineering.

What if the model author of the engine creates a digital engine that operates differently than the actual one? The automobile company could provide full access to an invalid model. We must have quality control measures in place to help us with this situation. The physical metaphor provides some help. Many consumer groups and institutions exist to protect consumers from bad products. Digital products will require similar groups and testing procedures. If a company knowingly markets a bad digital product, they will ultimately pay for this error in the marketplace. The digital object must be treated with the same level of quality control as the physical counterpart. In some cases, a company might make a mistake in production and a part or entire vehicle must be recalled. This type of recall is made easier with the digital product. It behooves the model authors to

create valid, quality objects. It may be that anyone can publish a digital object but this is true of physical objects as well. The situation is somewhat more acute with a digital automobile since to create an automobile in the first place, one must have invested a fair amount of time and resources; however, a digital engine could be created by the neighbor down the street. One must learn to trust certain sources more than others based on past performance of prior digital objects. Also, we must have ways of verifying our sources, developers and producers with methods such as digital signatures, watermarks and encryption.

3.3 Simulation Modeling Methodology and the WWW (Kevin J. Healy)

The World Wide Web was conceived as a set of simple Internet-based client/server protocols for transferring and rendering documents of a primarily textual nature. What distinguished the Web's mode of communicating information from other Internet-based tools that preceded it (e.g., electronic mail, electronic file transfer via ftp, and network newsgroups) was the provision for embedding hyperlinks that allowed users to easily navigate between related documents. The hyperlinking scheme allowed content providers to organize and present information in a natural hierarchical fashion. It also served to insulate users from the tedious details involved in identifying and retrieving a particular document. Since the development and rapid widespread adoption of these conventions, they have been extended and integrated with other new related technologies that provide for the delivery of content that is much more dynamic in nature. The most important of these related developments has been the introduction and rapid widespread adoption of the Java programming language as a standard for Internet-based computation.

The integration of the Web and Java represents a technological advancement that enables a fundamentally new approach to simulation modeling, one that makes possible the development of environments with coherent Web-based support for collaborative model development, dynamic multimedia-based documentation, as well as open widespread execution and investigative analysis of models. A key aspect to the approach is the role the Java language plays in both the specification and implementation of the model.

The evolution to high-level model specification languages and formalisms has been motivated by the desire to make simulation more accessible by eliminating the programming burden. However, such systems are often difficult to modify or extend because of an imposed separation between the specification system and its implementation. This can lead to models that poorly mirror system behavior and have no potential for distribution and reuse within an enterprise. The Java language is ideally suited to implementing an advanced simulation architecture whose features are readily accessible at the programming language level, special purpose simulation language level, and high-level model specifications. Specifically, key features like the well-designed object-oriented nature of Java and native

support for multithreaded execution allow special purpose simulation modeling features to be incorporated directly into the Java language in a natural way so that the underlying modeling and programming languages are the same. These relatively low-level but powerful modeling capabilities can in turn be used to implement higher-level model specification systems via the JavaBeans component development model. The simple programming conventions that constitute JavaBeans allow Java-based software components to be assembled visually into applications using any of a growing number of sophisticated graphical programming environments including Symantec's Visual Café, Microsoft's J++, IBM's Visual Age, Sun's Java Workshop, Borland's Jbuilder, and Lotus's BeanMachine. When visually assembling predefined simulation modeling components, no programming is required; however, when necessary, the user has access to the underlying code and full power and flexibility of the Java programming language. What's more, any Java environment can be used for model building and debugging. The modeling language capabilities and predefined component assembly capabilities can also be used in isolation or in combination to produce high-level standalone simulation applications that users interact with in predefined ways. The hardware and operating system independent design of Java facilitates collaboration by allowing modelers to share language level or component level models independent of where they were developed. The documentation and deployment of modeling tools and end-user applications via the Web also serves to make open and widespread both the development and investigative analysis of models. This vision of Web-based simulation is the motivation behind Thread Technologies' design of *Silk*™, a general purpose simulation language based on the Java programming language. *Silk* merges familiar process-oriented modeling structures with powerful object-oriented language features in an intelligent design that encourages model simplicity and reusability through the development and the visual assembly of *Silk* modeling components in JavaBeans-based graphical software environments. More generally, *Silk*'s openly extensible, scalable, and platform independent design represents the type of approach that is essential to keeping simulation modeling on track with other revolutionary changes taking place in Internet-based computing.

3.4 Simulation Modeling Methodology in the Wonderfully Webbed World (Richard E. Nance)

While modeling methodology has been with us since the inception of simulation, it remained indistinguishable from programming throughout the first two decades. Nevertheless, a few early researchers abstracted beyond the executable form to search for more significant semantic revelations. Lackner and Kribs [1964] andKiviat [1967] are prominent examples, but Tocher's [1966] wheel charts to assist in model specification and the IFIP proceedings on simulation programming languages Buxton [1968] shows that interest was widespread. Efforts to derive a theory of simulation

[Zeigler 1976] generated interest in model representation in the 1970s. The latter part of the decade ushered in the first specific focus on modeling methodology (model life cycle, model specification languages, the DELTA project) [Nance 1979]. With the 1980s came the vision of model development environments [Nance 1983] that are now a commercial reality. Is the subject of this panel session presaging the next major transition in simulation model development?

3.4.1 *Modeling Methodology*. Since "methodology" is both overused and misused, a definitional explanation in this context is appropriate. Methodology, following the view of Arthur et al. [1986], should:

—organize and structure the tasks comprising the effort to achieve global objectives,

—include methods and techniques for accomplishing individual tasks (within the framework of global objectives), and

—prescribe the manner in which certain classes of decisions are made and the ways of making those decisions lead to desired objectives.

Key in the attainment of the objectives are the *principles* that form the foundational support of a methodology.

3.4.2 *Influence of the Web*. If the World Wide Web is to effect major changes in modeling methodology, then it must alter or abolish existing principles or introduce new principles. At this juncture, the capability of the web to influence the technology of model building, model execution and model sharing is clear, and the degree of change appears significant. However, that the potential for influence extends into the principles—the foundational core—is less apparent.

3.5 Web-Based Simulation: Whither We Wander? (Ray J. Paul)

This panel contribution will discuss a variety of new technologies for software development and ways of working that will have an unpredictable effect on the future of simulation modelling.

3.5.1 *Multimedia/Synthetic Environments*. The ability to access multimedia on the Web clearly introduces greater potential for the use of videos of problem scenarios, for interaction with stake-holders situated at remote locations (for example, when the running model hits an unknown combination of circumstances, an expert stake-holder might be able to determine the successful rules for advance) and sound. For example, on a recent visit to a Hong Kong container terminal, I was shown a television control centre, computer-based, which had 100% video coverage of the terminal. While its purpose was clearly for security and safety, it requires little imagination to visualize how a simulation of the terminal operation could call up the appropriate video camera when problem discussants get to the point of a simulation run where clarification is desired. I think that the rush to join the much-hyped bandwagon of Synthetic Environments, driven by technical

extravagance and financial greed, is in great danger of neglecting or even forgetting those major simulation issues of ongoing concern over the years. These are the ongoing intractable problems of verification and validation. The current enthusiasm for Synthetic Environments is therefore in danger of creating more expensive mistakes, to the detriment of the reputations of simulationists, analysts and operations researchers in general.

3.5.2 *Natural Born Webbers*. A large proportion of the current generation of students entering higher education in the developed countries are already familiar with the pastime of browsing the Web and playing computer games. Both of these activities might loosely be depicted as approaches based on "suck it and see." Browsing and adventure games encourage the participant to try out alternatives with rapid feedback, avoiding the need to analyze a problem with a view to deriving the result. Such web users, in order to use simulation, need, and desire development tools that allow for fast model building and quick and easy experimentation. Furthermore, such web users should have a natural affinity to the use of simulation models as a problem understanding approach [Paul and Balmer 1993; Paul and Hlupic 1994]. Web-enabled simulation analysts will be opposed to classical software engineering approaches and methodologies. They will be seeking tools that will enable them to assemble rather than build a model. Some feel for the change of "culture" that we can expect from future generations of computer users can be gauged from a recent experience of mine on a visit to Taipei (Taiwan). A class of school children were using the local university's multimedia lab. A ten year old schoolboy was typing in HTML codes faster than I, and dynamically checking it by running a rather impressive text/video/sound demonstration system. The boy could not speak, read or write any English; everything was symbolic to him.

3.5.3 *New Software Technologies*. Some have predicted that the software industry will be divided into component factories where powerful and general components will be built and tested, and into component assembly shops where these components will be assembled into flexible business solutions. Such component-based development, if it occurs, might give significant gains in productivity and quality as well as known obvious benefits to web-based software development.

3.5.4 *Java*. Java is now so ubiquitous that it might appear unnecessary to comment on it. For completeness, the reader is reminded that simulation models in Java can be made widely available; an applet can be retrieved and run and does not have to be ported to a different platform or even recompiled or relinked; there is a high degree of dynamism because Java applets run on a browser; Java built-in threads make it easier to implement simulation following the process interactive paradigms; Java has built in supports for providing sophisticated animations and is smaller, cleaner, safer, and easier to learn than C++, allegedly.

3.5.5 *Conclusions*. For me, the aforegoing indicates a world of dynamic change, which I welcome, but where it is all going is a matter of conjecture that will be colored more by prejudice and opinion than evidence.

4. REACTIONS

In this section the authors respond to the points made in the previous section.

4.1 Ray Paul's Comments

Regarding the positions of Arnold Buss and Kevin Healy, it is arguable that Java is so good. We have experience of platform dependence, and of course the rate of enhanced releases which are not downward compatible outdates software rapidly. On the other hand, such fast adaptation of the language might encourage improved methods of release compatibility, to the benefit of the industry at large.

Regarding Paul Fishwick's position, it is arguable that quality control is necessary for software reuse. The traditional methods of building large models, which take much time and money, and which in themselves then lead to an expectation of repeated use, demand some sort of quality assurance. When it takes so long to get an answer(s), it is a bit limp to also admit that the model may be indeterminately wrong! However, if we can "glue" bits together fast and experimentally (Ray's crystal ball in action here), then maybe the emphasis will shift dramatically from "is the model correct?" to "is the analysis, albeit with unproven software, acceptable given the large experimentation that swift modelling has enabled us to carry out in a short space of time?" In other words, the search space has been dramatically reduced not by accuracy (the old way), but by massive and rapid search conducted by an empowered analyst (the new way).

Regarding Richard Nance's position, maybe our current principles are inappropriate for a web-based world. I have already argued some of this in the previous paragraph. Here I go further. We are in a period of rapid technical change (though some authors claim this will come to an end and life will settle down again— see Fernandez-Armesto [1995]). Every attempt we make to use these technological advances adds to the opening up of new opportunities to make change. This is particularly noticeable in business, where new companies are emerging fast, old ones are sinking daily, mergers, acquisitions, takeovers, etc. are prevalent. Even in the military sphere, the nature of the task to be faced changes quickly (war, peace-keeping, policing, training allies, reassessing threat as the political world moves on and so forth). Analysis needs to be fast, or else the problem has moved on anyway. Methods that produce ballpark estimates quickly, enhanced with more accurate methods if time allows, are or will be the order of the day. Principles based on output analysis, rather than modelling analysis, are likely to be more appropriate. If the traditional, analytical, and academic communities try to maintain current principles, they will

become historians, worthy of a footnote about Luddite Neanderthals in the next Millennium history.

4.2 Arnold Buss's Comments

Regarding Ray Paul's comments, he has indeed brought up some thought-provoking issues. With regard to Java, although it is good to be skeptical, it is clear at this point that the only thing that will derail its achieving true platform independence is willful destruction, to wit, Microsoft's attempts to make it Windows-specific. There is, in my opinion, simply too large a critical mass of developers and companies who are getting on board for this to happen.

Moreover, I believe that the Java component infrastructure (JavaBeans) will be precisely the platform on which to assemble large models from smaller components, so that the entire monolith does not have to be designed in one piece. I believe that component-based design will supplant OO design in a major way in the near future, in part fueled by network-based computing. On the network, you *must* be component-oriented or the thing is just too unwieldy. Designing distributed models in a reasonable manner pretty much forces you into components. The design issues focus more on responsibilities and interoperability rather than class hierarchies, as in traditional OO design.

There is a somewhat subtle aspect of the Java language that turns out to be the real winner for component-based design, namely interfaces (vice classes). Interfaces enable components to interoperate and pass messages *without having to know the precise class or class hierarchy of each other.* The interface is simply a contract to implement certain methods, so they may be invoked with compiler-safe impunity. Interfaces allow you to replace one object with another of an entirely different class with no necessary implied "is-a" relationship.

The second really important element is enabled by interfaces: communication via events. Interfaces allow you to define a small handful of event sources and event listeners that can provide communications between objects that is much more flexible than ordinary method invocation. One object will register its interest in another's events (or, more likely, be registered by a third party). Whenever the event source's state changes to trigger an event, all objects listening are notified. The key is that neither event sources nor listeners need to be "aware" that any of this is happening. Objects can register and un-register their interest as the program evolves. Event communication is a powerful means of implementing distributed models. Remote objects may easily register as listeners by using a remote mechanism (RMI, CORBA, etc.) and the event sources need not know (or care).

I have enhanced Simkit to incorporate this kind of messaging, and am currently working with a student to extend it further. For example, we have generic entities that are nothing more than containers. Functionality is put on these entities by creating and adding components. To enable

movement, for example, a Mover entity is thrown in. The kind of movement possible is entirely determined by the type of Mover. Add a sensor and you can detect other things (depending, of course, on the kind of sensor). If a Mover and a Sensor are put together in a container, then the movement is governed by the Mover. Basically, this is an extremely flexible type of composition. It is difficult to express in standard OO notation, since neither the Mover nor the Sensor are instance variables. Besides, Booch/UML diagrams just tend to confuse matters in my opinion. The point is that a generic component-based methodology needs to be developed for such modeling. Java is a perfect vehicle for doing just that.

4.3 Richard Nance's Comments

Regarding Ray Paul's comments, I do not accept the claim that "it is a bit limp to also admit that the model may be indeterminately wrong." A model is not reality and only a fool insists that a model be error-free (the same person who wants a world with no accidents). How do you propose to answer the shifted question above: "Is the analysis... acceptable?" I see your only recourse as an after-the-fact conclusion, which advances us to the stage of relying on prophets—why bother with the unproven software, etc? Having returned us to the technology of 2000 years ago, what next is to be offered? How about a roulette wheel with labeled outcomes?

Since the good Dr. Paul does not provide quotation marks or a page number, I assume that these sentiments are not those of Fernandez-Armesto but his own. My reaction is that I do not live in the fast lane that engulfs Dr. Paul. My long-time technical sponsor, the US Navy, is working now and for some three years prior on the design of the next destroyer (2003). The models and simulations used in this task are time-consuming to develop and the analysis is conducted over years, not days, hours or seconds. The hull will be in service for some 30 years and undergo three major overhauls, all of which requiring parts of the existing models and still others that will be developed, again perhaps in months, but certainly not in seconds.

I do not think our modeling methodology principles have been altered at all by the web. The capabilities of the tools based on the principles have changed and are changing, but that is the way technical progress is made.

My thanks to the good Dr. Paul for his agitating expressions of these misguided views. May he never fly on an aircraft developed with his analysis/prophet approach to decision making.

4.4 Paul Fishwick's Comments

Everyone on the panel makes valid points. There is a need to tie together some of the views to make a whole. At the same time, I'll express my own perspective on "where it all is going." Ernie Page's reference of Dijkstra's principle is one where we must separate specification from implementation. In general, this separation is one where we talk of "level of model." A piece of code, a mathematical expression, a Petri network, and a 3D cylinder are

all examples of models. We may choose one specific model to represent some aspect of the system that we are studying. The code may represent the same dynamics as the Petri net, while the cylinder may represent either an abstraction of the system shape or, perhaps, a state of the system. The interpretation that we foster is the essence of modeling. Modeling is an art in this respect.

Arnie Buss and Kevin Healy speak kindly of Java. Java does show promise for its intended function: a computer language meant to migrate over a large area network to promote distributed computing. At the same time, Java is a textual computer language, so its primary purpose is to represent dynamics at a fairly low level of abstraction. I'll submit that source code written in Java is a model, and that one can "think in Java" about system behavior. On the other hand, many people will not find this metaphor as appealing as one that is visual and graphical. Models must serve the user's view and way of thinking. There is no one correct modeling language. Ultimately, models are shared metaphors. If I am part of the Petri network or System Dynamics community, I think in these specific icons. The models color my thinking about dynamics. If we can all agree that we have many modeling types or languages, and that we can form translations among models (from Petri networks to Java, for example), then all forms of modeling become germane to the discussion. With Java at the lowest level of translation, our task of distributed execution of models is enhanced, and so research and development of Java is good for web-based modeling and simulation. Let's just keep in mind that Java is one of many nodes in a vast modeling network with models as nodes and translations as arcs. I know very few scientists or engineers who would prefer Java over their pet modeling methodologies.

Dick Nance and Ray Paul speak of two opposite poles in terms of quality in modeling. If I attend an art exhibition and buy a modern sculpture made of electronic home appliances—such as toasters, can openers, and mixers—I will most likely not use this artwork for engineering purposes. When special effects companies in Hollywood create models of the Titanic and of New York City, their objective is to foster entertainment and not to create statistically valid behavior. Therefore, both views are supported. Quality must be maintained where it is required, and to the degree that it is required depending on overall objectives of the simulation. There is nothing wrong with the Taiwanese schoolboy (Paul's example) who grabs objects left and right to create a new experience. Some of these objects, like the toaster in the sculpture, may be based on high resolution models (both structural and dynamic). It is the way in which the objects are used that determines the outcome, and all outcomes are fair game.

Luckily, the future is bright for simulation and web-based modeling and simulation. Imagine the Taiwanese schoolboy unchained from abstract languages such as HTML. Instead, a new world-wide marketplace of digital objects yields the digital equivalent of everything you see around you. The web is no longer fettered with documents. Documents are but one kind of physical object. The schoolboy will be creating complex games and simulations

for his friends who will later join him in a multiplayer extravaganza. Meanwhile, the Navy is testing out a new class of submarine using objects delivered by its contractors. This delivery occurs well before the physical submarine components are manufactured. Some of the Navy objects will be the same used by the schoolboy just as the toaster can be used in more than one way. The objectives of the Navy and schoolboy are different, but the digital object marketplace is common to both of them. I think that Dick Nance is right about a change in modeling methodology. It is happening now and web-based simulation is the catalyst. The purpose of physical objects is to achieve a singular objective, but the global objective is left open to the end-user. This is a departure from Arthur et al. [1986]. We should not limit our models by global objectives. Objectives and models are orthogonal. I use the web to locate objects, and I use these objects to create models of every variety. Like the manufacturer of the toaster, I create the best digital toaster possible and let the consumer make the choice as to its utility. I would not be at all surprised to go to Taiwan in ten years and find the schoolboy playing a "multiplayer deathmatch" inside the confines of a greatly enlarged toaster within a Daliesque landscape. Meanwhile, the Navy is modeling the high level dynamics of a towed-array sonar using a circuit of light bulbs from General Electric's web site, with bulbs representing states.

5. SUMMARY

The era of Web is certainly upon us. There seems to be no escaping that fact. The confluence of the Web and simulation offers an opportunity to change the way we approach modeling. How much should we embrace such change? The panelists disagree on this point. If the world of digital objects appears—and if publishing models on the Web becomes profitable, digital objects seem certain to proliferate—will the modeling process become enhanced or impaired? The act of model construction would be arguably simpler—assuming sufficiently powerful search engines. It should be much easier to "plug" models together than to build models from scratch. But then what? How will models be validated in this environment? In the absence of widespread adoption of open-source philosophies, model validation may become one big exercise in black-box testing. In areas where validation is critical, this situation can only represent a step backward.

But perhaps an engineering analogy is useful here. No one would argue, for example, that bridges are a bad idea. Although occasionally failures do occur (and such failures can be catastrophic), for the most part bridges are engineered for safety. Where possible, pathological situations are considered and accounted for in bridge design. Worst-case capacities (and then some) are accommodated. The opportunity to misapply the science and mathematics that support bridge design exists, but the engineering profession actively seeks to limit such opportunities.

Technology marches on. Modeling is central to technological advancement. But advancing technology impacts the modeling process as well. As

simulation becomes a desktop commodity, it will be available to masses. This ubiquity is a mixed blessing. Having access to such a powerful problem-solving technique is potentially quite valuable. On the other hand, to the untrained user—a user with a what-you-see-is-what-you-get perspective—the potential to misapply the technique is great. As responsible engineers of the future, should not those enabling the Web-based simulation revolution also shepherd the safety of the technique?

REFERENCES

ARTHUR, J. D., NANCE, R. E., AND HENRY, S. M. 1986. A procedural approach to evaluating software development methodologies and associated products. Technical Report SRC-87-007.

BUXTON, J., ED. 1968. *Proceedings of the IFIP Working Conference on Simulation Programming Languages*. North-Holland Publishing Co., Amsterdam, The Netherlands.

DIJKSTRA, E. W. 1976. *A Discipline of Programming*. Prentice Hall Press, Upper Saddle River, NJ.

ERKES, J. W., KENNY, K. B., LEWIS, J. W., SARACHAN, B. D., SOBOLEWSKI, M. W., AND SUM, R. N. 1996. Implementing shared manufacturing services on the World-Wide Web. *Commun. ACM 39*, 2 (Feb.), 34–45.

FERNANDEZ-ARMESTO, F. 1995. *Millennium: A History of the Last Thousand Years*. Touchstone, Inc., New York, NY.

FISHWICK, P. A., HILL, D. R. C., AND SMITH, R., EDS. 1998. *Proceedings of the 1998 International Conference on Web-Based Modeling and Simulation (Volume 30 of Simulation Series (Number 1))*. (Jan. 10 - 14). Society for Computer Simulation, San Diego, CA.

KIVIAT, P. J. 1967. Digital computer simulation: Modeling concepts. Tech. Rep. RM-5883-PR. The RAND Corporation, Santa Monica, CA.

LACKNER, M. R. AND KRIBS, P. 1964. Introduction to a calculus of change. Tech. Rep. TM-1750/000/01. System Development Corporation, Santa Monica, CA.

NANCE, R. E. 1979. Model representation in discrete event simulation: Prospects for developing documentation standards. In *Current Issues in Computer Simulation*, N. Adam and A. Dogramaci, Eds. Academic Press, Inc., New York, NY, 83–97.

NANCE, R. E. 1983. A tutorial view of simulation model development. In *Proceedings of the 1983 Winter Simulation Conference* (WSC '83, Washington, D.C., Dec.). ACM Press, New York, NY, 325–331.

PAGE, E. H., BUSS, A., FISHWICK, P. A., HEALY, K. J., NANCE, R. E., AND PAUL, R. J. 1998. The modeling methodological impacts of web-based simulation. In *Proceedings of the 1998 International Conference on Web-Based Modeling and Simulation (Volume 30 of Simulation Series (Number 1))* (Jan. 10 - 14), P. A. Fishwick, D. R. C. Hill, and R. Smith, Eds. Society for Computer Simulation, San Diego, CA, 123–128.

PAUL, R. J. AND HLUPIC, V. 1994. The CASM environment revisited. In *Proceedings of the 1994 Winter Simulation Conference* (WSC '94, Lake Buena Vista, FL, Dec. 11–14), M. S. Manivannan and J. D. Tew, Eds. Society for Computer Simulation, San Diego, CA, 641–648.

PAUL, R. J. AND BALMER, D. W. 1993. *Simulation Modelling*. Chartwell-Bratt, Bromley, UK.

SWARTOUT, W. AND BALZER, R. 1982. On the inevitable intertwining of specification and implementation. *Commun. ACM 25*, 7 (Jul.), 438–440.

TOCHER, K. D. T. 1966. Some techniques of model building. In *Proceedings of the IBM Scientific Symposium on Simulation Models and Gaming* (White Plains, NY). IBM Corp., Riverton, NJ, 119–155.

ZEIGLER, B. P. 1976. *Theory of Modelling and Simulation*. John Wiley and Sons, Inc., New York, NY.