Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

GPSS TURNS 40: SELECTED PERSPECTIVES

Thomas J. Schriber (Moderator)

Computer and Information Systems
The University of Michigan
Ann Arbor, MI 48104-1234, U.S.A.

Peter Lorenz

Institute for Simulation and Graphics
Otto von Guericke University of Magdeburg, PSF 4120
Magdeburg, Sachsen-Anhalt 39102 ,GERMANY

Springer Cox

Minuteman Software
P.O. Box 131
Holly Springs, NC 27540, U.S.A.

Julian Reitman

University of Connecticut - Stamford
One University Place
Stamford, CT 06901-2315, U.S.A.

James O. Henriksen

Wolverine Software Corporation
2111 Eisenhower Avenue, Suite 404
Alexandria, VA 22314-4679, U.S.A.

Ingolf Ståhl

Department of Managerial Economics, Box 6501
Stockholm School of Economics
Stockholm, SE - 113 83, SWEDEN

ABSTRACT

GPSS (General Purpose Simulation System) is celebrating its 40th birthday this year. We recognize this notable birthday by assembling a panel of discussants consisting of some of the folks who have contributed significantly to GPSS and its use over the years. The panelists are Springer Cox (GPSS/PC and GPSS World), Jim Henriksen (GPSS/H and Proof Animation), Peter Lorenz (promoter of GPSS in Europe and on the Web), Julian Reitman (principal in early interactive use and accommodation for large-scale simulations), and Ingolf Ståhl (micro-GPSS for Windows and on the Web), with Tom Schriber (author of the "Red Book") as moderator. Each panelist has contributed written perspectives describing aspects of his involvement with GPSS. A Geoffrey Gordon memoriam is included in the paper. (Geoffrey Gordon, who conceived and evolved the idea for GPSS and brought about its IBM implementations, died in 1989.)

1 INTRODUCTION

This introductory section provides a brief glimpse into the character of GPSS and the underlying synergies, motivations and objectives for bringing GPSS into existence. This glimpse takes the form of direct quotes from material written by Geoffrey Gordon, the developer of the original GPSS, and panelist Jim Henriksen. More details on the development of GPSS can be found in the Geoffrey Gordon memoriam further on in this paper.

Some twenty years after IBM released GPSS, and nine years after GPSS was ranked in the tenth position among what were then judged to be the world's thirteen most important programming languages (Sammet 1972), Geoffrey Gordon wrote a paper entitled "The Development of the General Purpose Simulation System (GPSS)." The four following paragraphs are taken from the first part of the paper (Gordon 1981):

"The General Purpose Simulation System (GPSS) is a programming system designed for the simulation of discrete systems. These are systems that can be modeled as a series of state changes that occur instantaneously, usually over a period of time. Complexities in their analysis arise because there are many elements in the system, and there is competition for limited system resources. The simulation technique uses numerical computation methods to follow the system elements through their changes of state, and predicts properties of the system from measurements on the model.

"GPSS came into existence rapidly, with virtually no planning, and surprisingly little effort. It came rapidly because it filled an urgent need that left little time for exploring alternatives. The lack of planning came from a happy coincidence of a solution meeting its problem at the right

time. The economy of effort was based on a background of experience in the type of application for which the language was designed, both on the part of the designer and the early users.

"Regarding my own background, I began simulating, with analog computers, in the early 1950s when working on guided missile studies at the Research Laboratories of the General Electric Company in England. Analog simulation is, of course, a different technique from digital simulation. However, no one who has worked with analog simulation can fail to have been impressed by the way an analog computer gets its user involved with the problem being solved. Putting together the elements of an analog computer to study a system feels almost like building the system itself, and it is extremely gratifying to get results that can be immediately related to elements of the system. There seem to be no intermediaries, no complicated procedures to follow, and no experts to interpret results.

"In developing GPSS there was no conscious effort to base the design on analog computers, but I feel sure the block diagram notation and the emphasis on making the simulation directly accessible to system analysts rather than through programmers, that are characteristics of GPSS, were unconsciously influenced by the analog computer experience."

And now, looking back over our collective shoulders at these developments that took place forty years ago, what do we see? Panelist Jim Henriksen sums it up very nicely with these observations (taken from his written statement):

"Gordon did one of the great packaging jobs of all time. He devised a set of building blocks that could be put together to build a flowchart that graphically depicted the operation of a system. Under this modeling paradigm, the flow of elements through a system was readily visible, because that was the focus of the whole approach.

"Gordon came up with a good solution to an important problem at a time when such a solution was desperately needed. Over the years that have ensued, transaction flow and variants thereof have become the dominant modeling paradigm in discrete event simulation. GPSS's transactions may be called *entities* or *items*, and GPSS's blocks may be called *nodes* in other software, but conceptually, there's a great family resemblance."

Amen!

2 PANEL MEMBERS' STATEMENTS

2.1 GPSS/PC^{tm}: A Personal Journey (Springer Cox)

My involvement with GPSS centers on our software company, Minuteman Software, our first product, GPSS/PC, and its successors. Throughout our involvement we have devoted more of our efforts to the development of the simulation environment than to the refinement of the GPSS language. The story deals more with the personal computer's influence on discrete event simulation than with the evolution of GPSS itself.

After working at IBM and Xerox as a computer performance analyst, in 1977 I began working in the Research and Development group at Digital Equipment Corporation for the purpose of simulating the performance of virtual memory operating systems. Over the next few years I created two simulation systems in Simula, the first major object oriented language. In both, I acquired experience in supporting a diverse customer base of simulation professionals. As a user of simulation myself in the analysis of computer performance, I ran into the frustrations that everyone had to tackle. On mainframes and minis, simulations were essentially done in a batch mode where you ran the simulation off-line, received the report much later, and only then could attempt to figure out what had happened.

Mainframe style simulation had a lot of inefficiencies. In batch mode you might be able to correct only a few errors at a time with each resubmittal of the batch job. I personally found this arms-length approach inefficient and frustrating, because it was extremely important to know what was happening inside the simulation and that processing was behaving as intended. Just as bad, it was nearly impossible to convince anyone else what was going on. Also, it was clear that users of mainframe-style simulation (like me) were wasting a lot of precious time that could have been better used in the design of the simulation and analysis of results. I had developed a small user base within DEC for my "SIMNET" program and had eventually devoted over half the Simula code to helping the user get the specifications of the simulation into the computer.

By 1978 I had a long list of features on my wish list. On top of the list was the desire for an interactive simulation environment that would protect me from my own mistakes. Ideally, I would be able to see what was going on inside the simulations. I needed to interact with the simulations in ways that could help me mold my intuition about the target system. Visual games were starting to appear on primitive microprocessor based systems, like "Asteroids" and "Star Wars." They clearly demonstrated how certain dynamics of simulated systems could be visualized. The next step, the application of interactivity and visualization to a commercial simulation package, was a "no-brainer," as they say. The 8-bit personal computers available at the time, like the Apple II, were interesting but had RAM of a maximum of 64k bytes or so -- clearly a fatal restriction as far as industrial simulations go. Even worse, it appeared that most people thought that since a discrete event simulation language could bring a mainframe to its knees if not properly tuned, there would be no way such a language could fit on a "toy" like a microprocessor.

Then in late 1979, Visicalc happened. The sales of the Apple II took off and IBM, by far the dominant computer company, reacted. Feeling the need to close Apple's window of opportunity as soon as possible (my opinion), they

commissioned a "quick and dirty" PC project. They wanted it within a year. That meant they would have to bypass IBM's normal product development procedure and use off-the-shelf hardware and software. They started shipping the IBM PC in 1981. It was an instant hit, selling over 10,000 units a month.

The IBM PC had one single characteristic that to me was a "go" signal: it had a 20-bit memory address bus. As far as I was concerned, the resulting 1MB address space solved the memory problem for a commercial simulation package. But what about simulation speed on a microprocessor? We needed to develop a prototype to verify feasibility.

An initial specification of the product had to come first. It was to be first-and-foremost an interactive simulation environment designed to take advantage of the unshared resources of a personal computer. Then, only secondarily, we had to choose a language. We wanted to implement a high level, visualizeable language with a large user base that we could sell into. The user interface had to be at a high enough level that users would be spared routine details and would be quick to develop simulations, but the language had to be complete in the sense that nearly anything could be simulated. The choice was easy. GPSS had a history of over 20 years with thousand of users, and several excellent textbooks.

By April of 1982 I had determined that the 6502 processor on an Atari 800 could run simple GPSS simulations at over 300 block entries per second. That's extremely slow by today's standards but back then it meant that many mainframe simulations could be run overnight on a personal computer. Also, by this time it was clear that the installed base of IBM PCs would eventually reach into the millions. There was no more time to waste; I began Minuteman Software and development on an IBM PC. I could not find a well-supported C compiler in early 1982, so I began to develop the user interface in compiled BASIC. However, since performance was an important issue, I coded the simulation paths in Intel 8088 Assembler Language.

Compatibility with existing GPSS Products was not strictly possible, but the closer GPSS/PC could be positioned to common GPSS experience, the better. I chose IBM's GPSS V as the standard. The language specification was based on IBM's GPSS V manual (IBM 1977). Tom Schriber's "Red Book" (Schriber 1974) and the GPSS V Book by Bobillier, Kahan, and Probst (Bobillier 1976) proved to be very helpful. I consulted the then-latest GPSS/H manual (Wolverine 1978) and one of Geoffrey Gordon's books (Gordon 1975) as well.

Without an existing product or customer base, I was able to engineer the product specifically for an unshared microprocessor with an online video monitor. Even in the 4.77-Megahertz Intel 8088 microprocessor there was an eternity of computer cycles between keystrokes that could be used for the user's benefit. In GPSS/PC I introduced a feature called Keystroke Error Prevention. By keeping an internal finite state machine representation of the GPSS grammar, each keystroke state transition could be tested for correctness, and rejected if in error. The net result was that in GPSS/PC it is impossible for the user to make a syntax error. Other features were directed at controlling and visualizing running simulations.

GPSS/PC followed GPSS V in using an integer clock. Although software emulation of real arithmetic was available, even those IBM PCs with a Floating-Point Processor (Intel 8087) ran slower with floating point calculations than with integer calculations. The implementation of a (nearly) unlimited precision integer clock would be fast and would solve several other problems as well. First, the 32 bit pseudo random number generators required a 64-bit product that was not supported by the hardware. Also, even with floating point arithmetic it was possible for the clock to grind to a halt because of loss of significance, and for overflows to invalidate report statistics and even the clock itself. The unlimited precision clock in GPSS/PC solved all these problems. The user needed only to declare a finer, less granular, time unit in order to increase overall precision. At the time I thought this to be such an improvement that I almost named the product "Precision GPSS" instead of GPSS/PC!

It took until 1984 to bring GPSS/PC to market (Minuteman 1984). Testing, documentation, and packaging all took time. On top of the list of new features was visualizeability. This design objective was one of the strongest reasons to get a simulation package onto the PC. GPSS/PC had windows on the major GPSS entity types built on a character graphics mode of the IBM PC. Within each window, up to 4 Microwindows could be opened which contained the values of the pre-defined internal state variables (Standard Numerical Attributes). All data on the screen were kept current with the changing state of running simulations. In the Tables Window one could observe the convergence of frequency distributions. In the Blocks window one could manipulate stop conditions and step commands from the keyboard, by using a mouse, or even a light pen. The user could set and remove stops by merely touching the appropriate block icon on the screen. As the simulation ran, the blocks changed appearance (highlighted or red) when congestion points appeared.

The product that we brought to market was a lot, but not all, of what I wanted. The report formatter was a separate program to save memory, and was unwieldy. The loading of a saved model into memory was slow because of the Keystroke Error Prevention, although since the Edit-Compile-Load-Run cycle was avoided, this was not so serious. The intent was that the user would only have to load once to set up the environment. Later modifications were to be done interactively with no need to reload. This was not much to pay for total protection from syntax errors.

In Version 2 of GPSS/PC, I replaced the compiled BASIC code with Lattice C. Alice (Cox) then joined the

company and proceeded to develop our Tutorial Manual for inclusion in the new release and to guide a repackaging of the product. The new GPSS/PC Tutorial took novice users into GPSS/PC simulation keystroke by keystroke. It also included the exploration of a set of very enlightening simulation applications that were developed by Professor Gerard F. Cummings of the University of Dublin (Minuteman 1986).

As our customer base increased, we attempted to respond to suggestions and requests. The GPSS/PC Animator appeared in 1988 (Minuteman 1988). It was a trace based post-processor implemented in a language called Autolisp. Using it, GPSS/PC simulations could be animated in a 2 1/2 D space created in AutoCAD. The Autoflix package, also from Autodesk, could then be used to create movies of the simulations.

By the late 1980s the IBM-compatible PC industry was shifting to Graphic User Interfaces akin to what had been available since 1984 on the Apple Macintosh. As competition intensified in the PC simulation software business, Minuteman Software began work on the new operating system developed and sanctioned by both Microsoft and IBM, OS/2. Our new product was named GPSS World^{tm} (Minuteman 1994) and required rewriting GPSS/PC in C++. It had many new features, but was most notably a distributed simulation system based on client/server architecture. Users could run simulations on a remote multitasked LAN server, but visualize and control them on their own PC's. Unfortunately for us, Microsoft then decided that Windows, not OS/2, would be the mainstream PC operating system. In response, IBM spent billions of dollars in an unsuccessful attempt to regain control of the industry.

After the failure of OS/2, we turned our attention to porting GPSS World to Microsoft's Windows operating system. During this process the user interface was rewritten and based on Microsoft's Document/View Architecture. The result was an object oriented user interface with inherited properties. Some of the technologies developed for OS/2 GPSS World such as address translation and multitasking were directly applicable in the new Windows-based product. Even the embedded programming language PLUS, which had been developed to beef up GPSS's powers of calculation, was extended and given the ability to control programmable experiments. The CONDUCT Command was then added to the lexicon of GPSS Commands in GPSS World for Windows, which was then launched in the summer of 2000 (Minuteman 2000).

Since then, development of new features has continued. In April 2001, we introduced the new Automatic Experiment Generators in GPSS World. In addition to a Multiway ANOVA routine to help analyze user-designed experiments, a Screening Experiment Generator and an Optimizing Experiment Generator have been integrated into the package. An Optimizing Experiment wanders over the experimental response surface utilizing a segmented steepest ascent search of up to 5 factors. The PLUS Experiment to perform the search is automatically created from dialog windows.

All of these features are supported by the Student Version. Unlike the Student Version of GPSS/PC, the Student Version of GPSS World runs just as fast as the Commercial Version--thousands of times faster on today's PCs than did the original GPSS/PC in 1984. Both Student Versions can be downloaded free of charge from our World Wide Web site: minutemansoftware.com.

Please pardon these commercial plugs, but don't overlook the fact that they demonstrate that GPSS is alive-and-well into the new millennium and is still under active development after 40 extremely productive years of service to humankind.

2.2 The Staying Power of GPSS (James O. Henriksen)

Let me say at the outset that it's a privilege to participate in this 40th birthday celebration for GPSS. In the vicious arena (!) of computer software, only a handful of languages of any kind have achieved GPSS's 40-year longevity. Why has GPSS survived so long? In the paragraphs that follow, I'll give my take on the contributing factors.

To start, we must consider how simulation was done before the release of Geoffrey Gordon's General Purpose System Simulator in October, 1961. Prior to that time, virtually all discrete event simulation was done using an event-based paradigm. Most simulation software included a central control routine that was variously referred to as a "timing routine," "simulation executive," "control program," etc. One described a system as a collection of events, and one wrote a "chunk" of code for each event. (Typically, this took the form of an *event routine*.) In discrete event simulation, time is viewed as a sequence of instants. The control routine was responsible for calling all events eligible to execute at any given instant and then progress to the next instant in simulated time. The next instant was the smallest future time in an event calendar for events scheduled to take place in the simulated future.

The event-based approach had one big advantage and one big disadvantage. The advantage was that it required no specialized language or operating system support. Event-based simulations could be implemented in procedural languages of even modest capability. Fortran was commonly used. The disadvantage of the event-based approach was that describing a system as a collection of events obscured any sense of process flow. For example, in a simple queuing model, for elements flowing through the system, one would describe events for arrival, start-of-service, end-of-service, and departure. In complex systems, the number of events grew to a point that following the be-

havior of an element flowing through the system became very difficult.

In the early 1960s, alternative approaches were available. The most obvious choice was some form of process interaction. Unfortunately, process interaction was understood only by an elite group of individuals and was beyond the reach of ordinary programmers. "Multi-threaded applications" were talked about in computer science classes, but rarely used in the broader community.

This was the primordial soup out of which the Gordon Simulator arose. Gordon's *transaction flow* world-view was a cleverly disguised form of process interaction that put the process interaction approach within the grasp of ordinary users. One of Gordon's objectives was to provide a tool that could, in fact, be used by non-programmers. It's interesting to note that forty years later, the debate as to whether simulation requires programming skills still rages on. Gordon did one of the great packaging jobs of all time. He devised a set of building blocks that could be put together to build a flowchart that graphically depicted the operation of a system. Under this modeling paradigm, the flow of elements through a system was readily visible, because that was the focus of the whole approach.

Gordon came up with a good solution to an important problem at a time when such a solution was desperately needed. Over the years that have ensued, transaction flow and variants thereof have become the dominant modeling paradigm in discrete event simulation. GPSS's transactions may be called *entities* or *items*, and GPSS's blocks may be called *nodes* in other software, but conceptually, there's a great family resemblance.

A number of other factors contributed to the longevity of GPSS. First and foremost was the emergence of a group of enthusiastic users who achieved success solving real-world problems. Among others, Julian Reitman's group at Norden had great success. Julian is fond of telling the following story. There once was a meeting at which a simulation project was discussed. After hearing the problem description, Julian said "We'll have Dick Baxter write a GPSS model." One of the meeting's other participants said "It can't be done in GPSS." Dick Baxter was not at the meeting, and no one bothered to tell him that the problem was impossible to solve in GPSS. Several weeks later, at a second meeting, Dick presented the results of a successful GPSS simulation.

Another significant contribution was the availability of textbooks describing GPSS. Although both Gordon (1968, 1975) and Reitman (1971) published books about GPSS, Tom Schriber's 1974 "Red Book" (so called because of its bright red cover) and its 1991 successor (Schriber 1991) became two of the best selling simulation textbooks of all time. In no small measure due to these books, GPSS was widely adopted in college courses in simulation.

Geoffrey Gordon's involvement with GPSS ended in the early 1970s. IBM (for whom Gordon worked) ended its support of GPSS in 1975, when GPSS was only fourteen years old. From that point forward, GPSS development and support was carried out by a small number of small, independent organizations. Saying these organizations were small is an understatement. The typical number of developers was one. For example, Springer Cox, Dave Martin, and Jim Henriksen each built commercially offered implementations of GPSS as 1-man efforts. By focusing on a single product line and offering capabilities that were attractive to users, the small firms were able to prosper.

Although GPSS has contributed greatly to the progress of the simulation community, in some respects it has begun to show its age. In order to discuss these respects, we must very carefully distinguish between those that are inherent conceptual properties of GPSS and those that are properties of its implementation. Conceptually, GPSS is very general. The fundamental building blocks of GPSS (generate, terminate, advance, seize, release, enter, leave, queue, and depart) have been incorporated in many variations in many simulation tools. It's hard to imagine any shockingly new queuing concepts that will arise in the near future that can't be modeled using GPSS's 40-year old concepts. The one disadvantage of GPSS's transaction flow world view (in my humble opinion) is that it overemphasizes the "active object, passive server" approach to modeling. In many systems, one must use an "active server" approach. While this can be done in GPSS, it's not the first approach that would occur to a beginning user. Notwithstanding the efforts of Tom Schriber, myself, and many others, the propensity for misuse still exists and is an inherent architectural property of the transaction world view.

The implementation of GPSS is really showing its age. In the original GPSS and for the most part, in all succeeding implementations, components of a model are structured as arrays, and individual components are accessed by specifying an *index* into an array. Faulty indexing is the number one source of computational errors in GPSS models. For small models, the use of integer indices is usual a manageable burden, although users must master knowledge of a basic set of quirks, or risk disaster. In very large models, the use of integer indices into arrays can become totally unwieldy. Large-scale logistics models often exhibit this property. In such models, data management is often a bigger problem than simulation, *per se*. Just keeping track of where everything is and what state it's in are major chores.

Wolverine's SLX (Wolverine 1996) has adopted much of the GPSS framework; however, SLX differs from GPSS in three respects. First, SLX implements GPSS constructs in a C-like, modern language framework. Thus, one can have arrays of servers, dynamically allocated servers, sets of servers, etc. Second, SLX is extensible, while GPSS is not. In SLX, one can devise building blocks of one's own choice that operate at the same level as GPSS's built-in, hard-wired blocks. Third, SLX exposes the underpinnings

of transaction flow and offers lower levels of description of parallelism than does GPSS.

Frequently I'm asked "Since you sell SLX, and SLX builds on GPSS/H (Wolverine 1978), why do you still sell GPSS/H?" The answer, in crass terms, is "People still want to buy it." In less crass terms, not every user needs the full-blown power of SLX. The prospective customer who needs to develop a straightforward queuing model, and who learned GPSS in school, is likely to be comfortable with GPSS/H and reluctant to move into "new territory."

Of the 40-year lifetime of GPSS, I have personally been involved with implementation of GPSS for 33 years. It's been quite a ride. Happy birthday, GPSS!

2.3 Perspectives from Germany (Peter Lorenz)

GPSS has been a formidable force in discrete-event simulation for 40 years. Let's look at some of the probable reasons for the longevity of GPSS, and speculate briefly on prospects for its continued longevity. My thoughts are stated in the series of points that follow.

2.3.1 Learning from Experience

GPSS was initially created in response to early experience in the formulation of simulation models (Gordon 1981). And in its evolution over time, GPSS has continued to reflect extensions of these experiences, combined with insights gained in practical applications. The GPSS entity classes and their associated methods are not just pie-in-the-sky inventions of a creative spirit. The most important entity class (transactions) and the other entity classes (e.g., facilities, storages, and logic switches) and their methods (blocks) are mappings into a simulation language of elements that appear in the real world of queuing networks and in other discrete systems. The fact that GPSS lends itself so readily and so well to the representation of discrete system reality, and did so early in the history of computing and simulation, speaks volumes for its longevity.

2.3.2 Gaining Friends Overseas

GPSS and its derivatives established themselves firmly in Germany (both East and West Germany) and more broadly in Europe (both Eastern Europe and Western Europe), and this helped increase the extent of its vigorous use and development. How did GPSS make its way to Europe? I believe it happened first through the printed word, followed closely by use of the corresponding software and then development of similar software patterned after GPSS, but often with enhancements. The earliest printed word to reach Europe probably took the form of the first GPSS User Manuals and associated documentation available in IBM computing installations in Western Europe, starting in the early to mid-1960s. (GPSS was released by IBM in October of 1961.) The practicality of GPSS-based simulation was the focus of the 1967 "Conference on the Applications of GPSS"; and the practicality of simulation in broader terms (not limited to GPSS) was then further demonstrated and documented at the annual successor conferences (now known as the Winter Simulation Conferences). Then came Julian Reitman's book, *Computer Simulation Applications* (Reitman 1971), which further legitimatised the practicality of simulation and stimulated the study and use of simulation in general and of GPSS in particular.

In Eastern Europe (including what was then East Germany), IBM-compatible computers were built beginning in the early 1970s. IBM's GPSS/360 and its successor, GPSS V, could be and were run on these machines. Special software for "mathematical methods" was also developed for these machines, and included SIMDIS, a discrete-event simulation language developed commercially as VOPS SIMDIS in Dresden in 1972-73, enhanced as PS SIMDIS in 1974, as SIMDIS-2 in 1982, and then further enhanced as SIMDIS-3 at the Otto von Guericke University of Magdeburg in 1987 (Preuss 1987). SIMDIS was patterned after the description of GPSS V in IBM's GPSS V Users Manual, but SIMDIS was more than just a re-implementation of GPSS V. SIMDIS extended the capabilities of GPSS V by inclusion of a database interface, for example, and by introduction of new blocks like MSELECT and MCOUNT (to carry out selection and counting operations within the rows and columns of matrices). In addition, SIMDIS-3 had interactive features.

The study and use of SIMDIS in particular and simulation in general was also stimulated in Germany and elsewhere through a German-language textbook (Frank and Lorenz 1979). SIMDIS saw considerable use during the 1970s and 1980s in East Germany and in the (then) Russian federation (including the Baltic states of Estonia, Latvia, and Lithuania) and in such (then) Soviet-bloc countries as Bulgaria, Czechoslovakia (now the Czech Republic and Slovakia), Hungary, Poland and Romania.

Why did the manufacturers of computing systems select GPSS as their language model, and not one of the alternative languages of that era (e.g., GASP, SIMSCRIPT, Simula)? There is a simple answer. GPSS had been identified in the early 1970s as one of the most important languages in the history of programming languages. In fact, GPSS was put into the tenth position in a list of the world's thirteen most important programming languages (Sammet 1972). This distinction was achieved during a period when hundreds of new languages and systems were being created annually in European and American universities. (The vast majority of these have long since fallen by the wayside.)

Also contributing to the spread of GPSS in Europe and elsewhere was Tom Schriber's "Red Book" (Schriber 1974). Many copies of that book (which went through about 40 printings in 20 years) found their way to Europe

and around the world. The Red Book also directly impacted the use of GPSS and its variants in the Russian federation and the Soviet-bloc countries in its Russian-language translation, the "Red Red Book" (Schriber 1980), of which 10,000 copies were printed.

That GPSS gained many adherents in Europe and elsewhere beyond the United States has been one of the factors contributing to its longevity.

2.3.3 Extension and Renewal

GPSS, the Grand Dame of simulation languages, has been the subject of extension and renewal on multiple occasions. Some of the key players in this regard are sitting with us as members of the panel. Seminal extensions into the arena of interactivity and incorporation of features facilitating the use of GPSS to model large-scale systems – that was done via Norden GPSS (in the 1960s time frame) under the leadership of Julian Reitman. Making GPSS fast and reliable and giving it its own control language – that was done via GPSS/H (with initial appearance in 1977) by Jim Henriksen. Providing GPSS to PC users and giving them new interactive interfaces and graphical output – that was done by Springer Cox via GPSS/PC (with initial appearance in 1984); Cox then went on to build a Windows-based GPSS World as well (with an OS/2 version in 1994 and a Windows version in 2000). Building a simplified and free version suitable for education – that was done by Ingolf Ståhl, in the form of micro-GPSS, with an initial international appearance in 1990 (Ståhl 1990), and of WebGPSS (webgpss.hk-r.se), first appearing in 1999 (Herper and Ståhl 1999). Players like Reitman, Henriksen, Cox and Ståhl have contributed substantially to the major enhancements of GPSS over the years that have supported its longevity!

2.3.4 Portability and Machine Independence

In the mid-1970s the idea of portability and machine independence became popular for software. The goals of portability and independence could be approached in the case of GPSS by using Fortran as a basic language for GPSS implementations. Such implementations were done in West Germany by Niemeyer (1972) and Schmidt (1979), and in East Germany by Lohse and Knocke (1989). These implementations freed "GPSS" from specific hardware to a considerable extent and made it available to a larger body of users.

Portability with *total* input and output fidelity has even been achieved for GPSS in terms of the complete correspondence of GPSS/H input files and output results, whether the GPSS/H models are run on mainframes, Solaris systems or DOS PCs. It was a real surprise for me to compare some 100 kilobytes of GPSS/H output from a PC-based simulation and a Unix-based simulation: the outputs were *exactly* identical – there was not one scintilla (or byte) of difference!

2.3.5 Ease of Model Enhancement

A common approach in developing a simulation application is to start with a simplified model and then add more and more detail over time. The specifications for a system and the model of the system often grow and evolve during the development of the application. GPSS lends itself nicely to this evolution.

Starting with simple examples and extending the examples step by step is also an approach commonly taken when teaching simulation. Again, GPSS supports this process admirably.

Is ease of model enhancement one reason for the position GPSS occupies in the teaching of simulation and for its survival in the applications of simulation?

2.3.6 Simple and Flexible I/O Files and Interfaces

GPSS input and output files are ASCII files. This promotes the ease of using these files as interfaces between other software and GPSS, and/or vice versa. An example is given below in terms of Proof Animation. Additional examples of such commercial interfacing in the case of GPSS/H can be found in Section 3 of Ståhl (2001). These features of simple and flexible GPSS input and output seem to be important for the continuing robustness of GPSS, too.

2.3.7 GPSS and Animation

The handshake between GPSS and animation has a long history. Reitman (1971, page 386) describes an early GPSS-based airport model (developed by Reitman's group at Norden) "made to draw pictures" that "dynamically change … as the simulation progresses." This is thought to be the first use of animation in a discrete-event simulation.

GPSS/PC (Cox 1984), SIMFOR (Lohse and Knocke 1989) and SIMPC (Schulze 1988) provide examples for combining GPSS with alpha-mosaic graphics. It is interesting that the referenced developments took place at a time when graphical displays were not yet standard computer components.

More recently, Proof Animation (Wolverine 1993) has been a significant development for the world of animation in general and for GPSS/H-based animations in particular. Proof Animation provides a significant example of how a GPSS/H output file can interface to other software. (Because ASCII files are the input to Proof Animation, any simulation or even non-simulation software that can write ASCII files can use Proof Animation to produce animations of the system being simulated.)

Animation plays a key role when I introduce students to simulation. My initial material makes use of Proof Animation as a simple simulation system with graphical output. The effect is to get the students very excited about and drawn into the study of simulation!

Furthermore, the fact that Proof Animation uses vector graphics is becoming of increasing importance at the University of Magdeburg. We hope to be able to create fast, high quality animations for the Web by implementing converters from Proof to alternative formats.

Animation in simulation is viewed nowadays in most quarters as a "must" part of commercial simulation. The fact that GPSS accommodates itself nicely to animation is another factor contributing to the longevity of GPSS.

2.3.8 Suitability for Teaching and Learning

As mentioned above, GPSS models can easily be extended. That is not the only advantage for the use of GPSS in teaching and learning. Other advantages include:

- the ability to avoid a "black box" approach when teaching simulation software. It is straightforward to bring students to an understanding of the internal logic and algorithms used by GPSS (Schriber 1991, and Schriber and Brunner 1998).
- the existence of many textbooks and the document "Simulation and Animation" that can be found at: http://www.isgsim1.cs.uni-magdeburg.de/~pelo/s1e/sim1.shtml.
- the absence of complicated object-class structures
- the simple interface to Proof Animation, which is easy to understand, learn and use.

Are these the reasons why, as of a survey taken in 1997 (Reinhard 1997), GPSS is the most frequently used language in simulation courses offered in universities in Germany, Austria, and the German-speaking part of Switzerland? (The survey shows that GPSS is used in 20 courses; Simula in 16; etc. See the reference for details.)

2.3.9 The Future

Let me finish with a personal look at just one aspect of future possibilities for GPSS. This aspect involves the potential for GPSS in terms of the World Wide Web.

GPSS was established in 1996 as one of the first simulators available in the Web (Lorenz et al., 1997). Simulations can be performed on our Magdeburg Web pages and animations can be viewed on our Web pages, too. Follow the B2B Simulation Initiative at http://www.b2bsim.de/ and see what happens. Much of what you see is based on GPSS, and some things you see are supported by Proof Animation. (Note, however, that the B2B Simulation Initiative is open for all languages and systems; it is not limited to GPSS.)

These developments point to the possibility that GPSS might become one of the pioneers in establishing a Simulation Service Provider (SSP).

Based on these Web developments alone, I believe that GPSS in its current forms is robust enough not only to have survived the person who conceived it, Geoffrey Gordon (Gordon 1981), but also to survive those among his successors who are sitting here in this panel.

2.4 The First Ten Years of GPSS (Julian Reitman)

There are two ways to review the history of GPSS, either from a language viewpoint or from the viewpoint of user accomplishments. My training as an engineer has biased me toward the latter. Therefore, my view of GPSS history starts with the breakthrough produced by GPSS I. Simulation, for the first time, could be timely, useful and economical. A new world of capability had emerged from Geoff Gordon's efforts. He provided the basic structure, which allowed the enhancements that followed. Later versions overcame some of the initial limitations. Our modeling efforts succeeded to a greater degree than could have been predicted. Details are provided below.

My entry into simulation started when I joined the Teleregister Corporation in 1955. I had entered the world of real time systems. The approach to designing real time systems was then in its infancy. Airline passenger seat reservations relied on posting on a board the status of each flight for the reservations telephone agent to scan visually. The Teleregister system provided each agent, local and remote, with a terminal that presented seat inventory status in response to a specific interrogation. The critical systems problem was the classic - what is the peak period traffic that the communications/computer system must handle. There was a long history of analysis of telephone systems traffic, but we discovered to our dismay that real time airline reservations traffic was not like telephone traffic. Agent's peak period actions were not independent transactions. The demand was hugely in excess of our predictions. Finally, the suggestion arose to use the Rand Corporation's "Table of one Million Random Numbers" in the context of discrete event simulation. A desk calculator and one week of effort produced one simulation result and the knowledge that manual simulation was impractical for commercial use. Switching to an IBM 650 showed that computers were difficult to program and needed more memory and that debugging took unaffordable time.

In 1961 I joined the Norden Division of United Aircraft, now United Technologies, to design a real time system to provide nationwide aviation weather data. IBM and others were visited to assess real time hardware. When questioned about subsystem performance, IBM explained

that the subsystem operation had been simulated. Instead of describing the simulation, IBM responded that it was company proprietary. Then something changed at IBM and there was an October 3, 1961 briefing on the "Gordon Simulator" (which was soon to be re-named GPSS I).

In preparation for that meeting, I reviewed my simulation experiences at Teleregister and prepared a list of what a simulation system should be able to do, listing thirteen capabilities. GPSS I handled all but one of these: it lacked the ability to identify a transaction and place it on what became in GPSS III a user chain controlled by LINK and UNLINK blocks. After Geoffrey Gordon's GPSS presentation at the Eastern Joint Computer Conference in Washington in December 1961 (Gordon 1961), training in GPSS was provided by IBM in early 1962. The punched cards for the GPSS system were installed at the United Aircraft Corporation Research Center. In those simple days we got the deck and documentation with no paper work, no fee, and not even a proprietary statement.

Norden's first use of GPSS I in 1962 was to determine the frequency of interference on a communications line when two terminals bid for line control at the same instant. The experts were of two camps - "no problem" and "won't work." Who was right? For the distribution of expected interarrival times, mathematical approaches were not useful. The surprise was the speed and ease of setting up the GPSS blocks, building, running, and debugging the model. The results were an indication of the value of simulation. Instead of "no problem" and "won't work," we got an unanticipated result, somewhere in between. The lesson was significant. We obtained unexpected results that forced us to reexamine our basic approach and to get additional data. Our insight had grown. Compared with previous simulation experiences, only a remarkably small effort, about two weeks, was needed to create the model, debug it, and develop confidence in the results.

Next, Norden management supported an in-plant course on company time to spread the use of simulation. One early model, "Modeling as Applied to the Evaluation of Alternative Systems" was presented at the 1964 IEEE Systems Science Conference held at the University of Pennsylvania.

An early experience with Sikorsky Aircraft showed a different side of the value of simulation. The problem was maintenance strategy for engine gearbox combinations for a twin engine aircraft. Engines would be removed for periodic maintenance or in the event of failure. Removal of the engine and gearbox combination was faster. However, a number of good gearboxes would have to be serviced. The simulation model was expected to indicate the better alternative. The concerned engineers carefully reviewed the model. As the model logic was checked out there was disagreement among the engineers as to the maintenance procedure. The model had become the means to force the groups to agree on a common system definition. Igor Sikorsky was briefed on the simulation approach. Sikorsky was already quite elderly at the time, but he was one who quickly saw the potential for the simulation approach and supported its use.

Norden management supported developing simulation models for organizations outside United Aircraft. In time the simulation group under my direction reached eleven people. The first model for an outside organization, the Coast Guard, analyzed the cost effectiveness of different approaches to the maintenance of the SPN-39 Loran C Receiver. Then, for the United States Navy Applied Science Laboratory, GPSS III was used in 1964-65 to model the "Cost Effectiveness Trade-Off of a Microcircuit Shipboard Display System." The experience gained from these efforts led to a significant Norden contribution, that of developing the capability to model large, complex systems. The Navy needed to compare different sets of equipment for a future ship and predict logistic and maintenance performance. IBM had introduced GPSS/360 in 1967 with the added feature of storing a data base in array form. Such a huge model was beyond the capability of GPSS/360, which required all blocks and data arrays to be resident in memory. Norden, with the cooperation of the IBM GPSS team, John Bult and Bob Gould in particular, undertook the task of adding to GPSS the ability to store both data arrays and model segments on disk and bring them into the memory partition as needed. It is noteworthy that back then, mainframe memory available to the user might be "as much" as 256K or as little as 92K bytes in size. In practical terms, models of unlimited size and complexity could run, even if slowly.

The Navy prepared dataset card decks using 026 IBM card punches while the System 360 used 029s. Sometimes this led to problems. Each run took an hour using the full memory of an IBM 360/50 computer. An input data error could cause the model to loop, but there was no way to check for such looping. We remedied this situation at Norden by incorporation of an IBM 2250 display into GPSS. Using this display, we could interrupt model execution and display the model state. (This was the first interactive use of GPSS.) If all was as expected, the run was continued. In addition to finding punched card errors, the 2250 reduced debugging time.

The "Conference on Applications of Simulation using GPSS" in 1967 was a seminal event. Although there were no printed proceedings, the large attendance, over 400, (including seven presenters from the Norden group), spread a sense of simulation's possibilities. The Norden modifications to GPSS were presented by Hunter et al.: "The Use of Disk Storage to Expand the Size Capabilities of GPSS." One operational Norden model was Baxter's "Prediction of a Naval Vessel's Performance." An extremely advanced model for the period was Fenn's evaluation of the performance of a coordinated group of helicopters in anti-submarine activity against a number of submarines attack-

ing a surface convoy, an "Antisubmarine Warfare Game." What is probably the longest-lasting model in terms of its growth and usage for more than twenty years was first presented in Seidler's "Simulation of Built-in Test Effectiveness of Airborne Radar."

The 1968 "Second Conference on Applications of Simulation" included a digest of the papers. The Norden group provided papers describing a very complex system (Ingerman 1968) and the details of the use of the 2250 display system (Hunter and Reitman 1968).

In 1970 Norden made four versions of GPSS/Norden available to outside users. Remote users throughout the world were able to use GPSS/Norden via the computer network of NCSS as well. In addition, the Norden simulation group used the skills it had developed to produce a version of GPSS for the CDC 6000.

The first ten years built confidence that complex systems could be successfully modeled. A community of simulation users had emerged. The first ten years also showed the difficulty in getting decision makers to accept the potential for simulation. At Norden, development efforts became more and more limited in scope because of the dictates of management. GPSS/H then eventually appeared, and the Norden simulation group switched to it.

2.5 The "Red Book" (Thomas J. Schriber)

Half way through my first year (1966-67) as an Assistant Professor on the faculty at the University of Michigan, my department chair invited me to develop an elective course on discrete-event simulation for the MBA curriculum. Surveying the language scene, it was easy to conclude that GPSS might be the best choice of language for such a course. Why? Because it had the merits of being sparse in its syntax, and of letting the model builder think quite directly in terms of the elements of the system for which a model was being built. (It would not be necessary for the students to learn the syntax and semantics of a programming language and deal with the coding of event routines and the like, which would almost be guaranteed to lead to low enrollments in the MBA environment, and perhaps now in many other environments, too, in the point-and-click, drag-and-drop world of today.)

And so, in the summer of 1967, I scrambled to start to master the details of GPSS myself and fashioned a syllabus for the first course offering, which then took place in the fall of 1967.

The GPSS literature available was quite thin, so I began developing my own notes and examples, often only one class ahead of the students. That was in the days when it was typical to lecture at the blackboard. But that meant you couldn't cover much material in class. (It took too long to draw block diagrams on the blackboard.) So I began to write out lectures ahead of time on transparencies and used a projector to move the material faster. The students couldn't write quickly enough to keep up with me, so after each class copies were made of the transparencies to give out at the next class. This was inefficient, but it worked as a stopgap measure for the first two course offerings.

As an aside, Jim Henriksen was a student in Michigan's MBA program at the time, and took the second (Winter 1968) offering of the simulation course. Jim sat quietly in the back row and said nothing for the first several weeks of the course. I thought, "Who is that guy back there anyway?" One day Jim then charged onto my radar screen with a penetrating question that gave me a lot of insight into "that guy." (As it turned out, Jim had a part time job in those days, too, maintaining simulation software in the University of Michigan system. So in our relationship, Jim came to me hat in hand (sort of) as a student, and I got in touch with him hat in hand (most definitely) when bugs surfaced in GPSS/360. When we ran into bugs, Jim was the guy who jumped into the GPSS/360 assembly language code to find and fix the bugs. About 30 bugs were found and fixed at Michigan in 1968-69!)

In the summer of 1968, just after Jim took the course, I revised my lecture materials and contracted with a local printer to have them printed in softbound form. The resulting "book" was put on consignment at a local bookstore for use in the 1968-69 offerings of the course. Representatives from several publishers saw the "book" and offered to publish it in a formalized version. After several more "preliminary printings" were tested at Michigan, the "Red Book" hit the streets in 1974.

The "Red Book" seemed to fill a gap in the literature, and took off. Later, when people asked why there wasn't yet a second edition of it, the answer was something like "Why work on a second edition now? The first edition is doing well, and there are so many other demands on one's scarce time."

Eventually that 1968 student, Jim Henriksen, came out (in 1977) with his own mainframe GPSS/H, and in due course it migrated to the desktop. Here was a compelling reason to write another GPSS book, this one on GPSS/H (Schriber 1991). To keep the price down, the publisher set a page limit of 400 pages. GPSS/H was far too rich to be covered with abundant detailed examples in 400 pages, so the 1991 book turned out to be only introductory in nature, even though plans had been in place to include the comprehensive treatment that GPSS/H was being given in courses at Michigan. Even today, when a copy of the "Red Book" comes into view, I sometimes find my thoughts moving in the direction of a "Son of Red Book"…

2.6 GPSS – 40 Years of Development (Ingolf Ståhl)

The views of Ingolf Ståhl are included in his paper "GPSS – 40 years of development," which appears elsewhere in these proceedings. See especially Section 6 of that paper.

3 GEOFFREY GORDON – IN MEMORIAM

Much of this material has been taken from Gordon (1981).

Geoffrey Gordon was born in England May 17, 1924. He began simulating with analog computers there in the early 1950s, and with digital computers in 1954. Coming to the United States in 1955, he continued digital simulation at the Westinghouse Corporation. In late 1956 he joined the Bell Telephone Laboratories, where he eventually began writing simulation programs for message switching systems. It became apparent that the individual items of equipment could be represented as simple server units, with their own service discipline and capacity. The system models were essentially networks of such units.

Gordon was then asked, in 1959, to work on a project at Bell Labs to develop a tool for studying advanced switching system designs. The project was based on sequence diagrams, which use graphs whose nodes correspond to operations and whose directed paths of connected nodes represent possible sequences of events. Gordon cowrote a Sequence Diagram Simulator and got it running by the end of 1959.

In 1960 Gordon joined the Advanced Systems Development Division of IBM. The division was heavily engaged in the design of teleprocessing systems, and the potential value of simulation in providing realistic models was recognized. Gordon suggested developing a system description language based on the Sequence Diagram Simulator approach. Rapid progress was made, and he soon began writing a program implementing a corresponding block diagram language. He was the only programmer, but worked closely with others who were involved in system designs and were active in assessing the developing program and suggesting improvements.

In addition to being able to describe many types of systems, the block diagram language provided an excellent means of communication. When Gordon talked with a system analyst, the two could quickly agree on a block diagram to describe the main elements of a system. By progressively refining the block diagram, the details of the system description could be expanded. Things evolved to the point that Gordon would produce a block diagram, working with someone familiar with the system to be simulated, and that person would then learn enough about the program to run it and make changes and extensions.

Existence of the program became well known within IBM. Gordon decided to document what had been produced to satisfy the growing internal demand for the program. The result was a cleaned-up program with a user's manual, produced as an IBM confidential document dated October 25, 1960. The program had never been given a name; by default, it came to be known as the Gordon Simulator.

At the beginning of 1961, a complete rewrite of the program was begun by Gordon and two co-workers. The program was brought to the attention of the Cross Industry Marketing Group of IBM, which agreed to sponsor it as an IBM product. The program's confidential classification was removed early in 1961. On October 6, 1961, the program was made available outside of IBM for use on the IBM 704, 709, and 7090 systems as GPSS I.

The original GPSS and its later versions went on to become very widely used. And Geoffrey Gordon went on to become well known, not only as the originator of GPSS but also as the author of two textbooks on the topic of system simulation (Gordon 1968; Gordon 1975), as well as for his contributions to encyclopedias and handbooks on computer science and operations research.

Geoffrey Gordon finished his career with IBM as a Consulting Systems Engineer and an IBM Fellow. After retiring, he taught for several years at Kean University in Union, New Jersey. Geoffrey Gordon died at age 65 in Washington, New Jersey, on December 19, 1989.

ACKNOWLEDGMENTS

We thank Alice J. Cox (Minuteman Software), who ferreted out valuable information about Geoffrey Gordon, and Robert Melworm (Kean University of New Jersey), who gave Alice useful facts. And we thank Robert G. Sargent (Syracuse University) who first pointed out that GPSS would "turn 40" in 2001, suggested that this fortieth birthday should be celebrated, and provided insightful commentary after reading drafts of this material.

REFERENCES

Note: To save space, some panelist references are listed in Ståhl (2001). Please refer to Ståhl (2001) for them.

Gordon, G. 1981. The development of the general purpose simulation system (GPSS). In *History of Programming Languages.* New York: ACM.

Hunter, S. and J. Reitman 1968. GPSS/360-Norden, a partial conversational GPSS. In *Proceedings of the Second Conference on Applications of Simulation*. Piscataway, New Jersey: IEEE.

IBM. 1977. *General Purpose Simulation System V Users Manual*. White Plains, New York: IBM Corporation.

Ingerman, D. 1968 Simulation of a railed automated highway. In *Proceedings of the Second Conference on Applications of Simulation.* Piscataway, NJ: IEEE.

Lohse, K. and R. Knocke. 1989. SIMFOR–ein allgemeines Simulationssystem für PC. In *Wissenschaftliche Mitteilungen des VEB FER*. Magdeburg: VEB FER

Lorenz, P., H. Dorwarth, K.-C. Ritter, and T. J. Schriber. 1997. Towards a web based simulation environment. In *Proceedings of the 1997 Winter Simulation Conference*, ed. S. Andradottir, K. J. Healy, D. H. Withers, and B. L. Nelson. Piscataway, NJ: IEEE.

Minuteman Software. 1984. *GPSS/PC Reference Manual*. Holly Springs NC: Minuteman Software.

Minuteman Software. 1986. *GPSS/PC Tutorial Manual*. Holly Springs NC: Minuteman Software.

Minuteman Software. 1988. *The GPSS/PC Animator Manual*. Holly Springs NC: Minuteman Software.

Minuteman Software. 1994. *OS/2 GPSS World Reference Manual*. Holly Springs NC: Minuteman Software.

Minuteman Software. 2000. *GPSS World Reference Manual*. Holly Springs NC: Minuteman Software.

Niemeyer, G. 1972. *Die Simulation von Systemabläufen mit Hilfe von FORTRAN: GPSS auf FORTAN-Basis*. Berlin-New York: Walter de Gruyter.

Reinhard, A. 1997. ASIM Umfrage: Simulation in der Lehre. Responses to this survey and can be found at <http://www.fps.maschinenbau.uni-kassel.de/Forschung/Fabriksimulation/Sim_i_d_lehre/sim_i_d_lehre.htm>.

Reitman, J. 1971. *Computer Simulation Applications*. New York: Wiley Interscience.

Sammett, J. E. 1972. Programming languages: history and future. *Communications of the ACM*, 15, 601-611. New York: Association of Computing Machinery.

Schriber, T. J. 1980. *Simulation Using* GPSS (Russian language edition; translated by Professor Michael Feinberg). Moscow: Mashinostroyenie Press.

Schulze, T. 1988. SIMPC – an implementation of GPSS for personal computers. In *Systems analysis and* simulation, ed. A. Sydow. Berlin: Akademie-Verlag.

Ståhl, I. 1990. Introduction to Simulation with GPSS: On the PC, Macintosh and VAX. Hemel Hempstead, United Kingdom: Prentice Hall International.

Ståhl, I. 2001. GPSS – 40 years of development. In *Proceedings of the 2001 Winter Simulation Conference*, ed B. A. Peters et al. Piscataway, New Jersey: IEEE.

Wolverine Software Corporation. 1978. *GPSS/H Users Manual, 1st* Edition. Alexandria Virginia: Wolverine Software Corporation. (Editor's note: the 1978 reference is given for historical accuracy. The manual is now in its third edition.)

Wolverine Software Corporation. 1993. *Using Proof Animation*. Alexandria, Virginia: Wolverine Software Corporation.

Wolverine Software Corporation. 1996. *SLX: An introduction for GPSS/H users.* Alexandria, Virginia: Wolverine Software Corporation.

AUTHOR BIOGRAPHIES

THOMAS J. SCHRIBER is a Professor in Computer and Information Systems at The University of Michigan. He has worked in the area of discrete event simulation since 1967, and has been active in the Winter Simulation Conferences from 1968 forward. He received the INFORMS/CS Distinguished Service Award in 1996.

PETER LORENZ is Professor Emeritus in the Institute for Simulation and Graphics at the Otto von Guericke University of Magdeburg in the state of Sachsen-Anhalt, Germany. He continues to teach discrete event simulation and animation there, and consults as well. His research interests include layout-based simulation-model generation, advanced Web-supported teaching concepts, applications of simulation and animation in mining, manufacturing, logistics and traffic, and moving simulation and animation to the Web. In 2001 he founded the B2B Simulation Initiative.

SPRINGER COX received his degrees in physics and computer science from Cornell and Syracuse University, respectively. He worked in computer performance evaluation and modeling for IBM, Xerox, and DEC. In 1982, he founded Minuteman Software for the purpose of creating a microprocessor based interactive simulation environment. He has published over a dozen technical papers and has spoken at technical conferences in North America and Europe.

JULIAN REITMAN retired from a 26 year career in the Norden Division of United Technologies in 1987. He helped instigate IBM's 1961 release of GPSS, and was instrumental in organizing the 1967 "Conference on the Applications of GPSS," for which he was the Program Chair. In 1968 he was General Chair of the successor "Conference on the Applications of Simulation." His sustained contributions to the area of simulation were recognized in 1998, when he received the first-ever INFORMS College of Simulation's Professional Lifetime Achievement Award. In retirement, he pursues an avocation to document the history of technology, developing and teaching a course on that topic these past 14 years at the University of Connecticut at Stamford.

JAMES O. HENRIKSEN is the president of Wolverine Software Corporation. He was the chief developer of the first version of GPSS/H, of Proof Animation, and of SLX. He is a frequent contributor to the literature on simulation and has presented many papers at the Winter Simulation Conference. Mr. Henriksen has served as the Business Chair and General Chair of past Winter Simulation Conferences. He has also served on the Board of Directors of the conference as the ACM/SIGSIM representative.

INGOLF STÅHL is a Professor at the Stockholm School of Economics in Stockholm, Sweden, in Computer-Based Applications of Economic Theory. He has taught GPSS for twenty-five years at universities in Sweden and the USA. Based on this experience, he has led the development of the micro-GPSS system.