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

TWENTY-FIFTH ANNIVERSARY PANEL DISCUSSION

THE WINTER SIMULATION CONFERENCE: PERSPECTIVES OF THE FOUNDING FATHERS

Michel Araten

The Chase Manhattan Bank
101 Park Avenue
New York, New York 10178, U.S.A.

Harold G. Hixson

U.S. Air Force Materiel Command/XPS
Wright-Patterson Air Force Base
Ohio 45433, U.S.A.

Austin C. Hoggatt

School of Business Administration
University of California, Berkeley
Berkeley, California 94720, U.S.A.

Philip J. Kiviat

KnowledgeWare, Incorporated
1650 Tyson's Boulevard, Suite 800
McLean, Virginia 22102, U.S.A.

Michael F. Morris

Productivity Management Center
6714 Whittier Avenue
McLean, Virginia 22101, U.S.A.

Arnold Ockene

IBM Corporation
472 Wheelers Farms Road
Milford, Connecticut 06460, U.S.A.

Julian Reitman

Department of History
University of Connecticut, Stamford
Stamford, Connecticut 06903, U.S.A.

Joseph M. Sussman

Department of Civil Engineering
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139, U.S.A.

James R. Wilson (Moderator)

Department of Industrial Engineering
North Carolina State University
Raleigh, North Carolina 27695, U.S.A.

ABSTRACT

In this paper the General Chairs and Program Chairs of the Winter Simulation Conference (WSC) during the period 1967–1974 continue the discussion started in the Keynote Address on the past, present, and future of WSC and the field of simulation.

MICHEL ARATEN

SIMULATION: BACK TO THE FUTURE

Like me, you're probably wondering what I am I doing here, since I haven't actively contributed to the practice of simulation for over seventeen years. I am a Vice President at the Chase Manhattan Bank, responsible for coming up with creative solutions to the real estate finance problems that some of you may have heard about.

It turns out that I was Program Chair and General Chair of the fourth and fifth Conferences, respectively. My paper was one of seventeen presented at the first Conference on Applications of Simulation Using GPSS held twenty-five years ago. The title was "Company Model for Income Prediction," and that's where my story and topic start. The paper described a project wherein management at Celanese Chemical wanted to improve their next year's earnings forecast by developing a probability distribution

of income. We approached this by interviewing managers to obtain subjective probability distributions for factors important to earnings—prices, volumes, costs, manufacturing capacities, and constraints—over five hundred distributions. These distributions were thrown into a GPSS model that acted like a crude linear-programming/materials-balance model and convoluted the distributions into an overall corporate-earnings probability distribution.

What was of interest to me was that a review a year later showed that almost half of the actual values of the individual variables fell outside the tails of their originally assessed distributions. Were these flaws a function of the interviewing techniques, lack of knowledge of probability theory by the estimators, or just inherent in human nature? The use of subjective probability theory in capital-investment risk analysis was also madly popular in corporations, and I even put together a GPSS model that was used by IBM as a tutorial. If there was a fatal flaw in the ability to assess subjective distributions, then much of the applications of these decision models would have reduced value.

This was a perfect doctoral-dissertation topic, but I needed some human guinea pigs to do my tests—first give them a short seminar on subjective probability theory, then provide them with standard scenarios, elicit their probability assessments, and finally compare the results for veridicality and consistency. I already had a group of assessors who knew little probability theory—the ten corporate managers who originally piqued my interest. The twenty undergraduate operations-research students I was teaching at Columbia made up the second group—probability theorists who had little practice in being assessed. To round it out I needed a group of practicing professional modelers who, similar to me, found themselves in the role of soliciting probability distributions for input into simulation models.

As you might imagine, I didn't have to search far. At the third Conference held in Los Angeles in 1969, I found my third group. Using the carrot of conducting a seminar on subjective probability assessment, I got thirty conference attendees to spend a couple of hours filling out probability distributions involving nine different scenarios. I still retain the mental vision of your colleagues feverishly working away on my scenarios, like college students taking final exams.

The result—aside from a doctorate—was the conclusion that, regardless of group background, while individuals did not differ significantly in assessing means, they did not agree on the shape and variance of the underlying distributions. While they managed to judge higher variability for predicting events further out in time, individuals could not easily combine assessments of ostensibly independent variables and were not efficient Bayesian utilizers of sample information.

Clearly, the subjective probability input into the simulation models was highly suspect; and the reliance on the resulting model output was troublesome. The whole impetus for probabilistic assessment was a reaction to both single-point (subjective) forecasts and to sensitivity analysis, where ranges of outcome variables are identified, but no one has a clue as to their relative likelihood.

I imagine that other practitioners, over time, came to similar conclusions—why spend a lot of effort collecting subjective probability distributions of input variables if the result is not demonstrably more reliable than single-point estimates? In particular, having a computer-generated probability distribution of an outcome variable can lead to a false sense of confidence by the decision maker.

Rather than throw out the baby with the bath water, in my professional work, I continued to elicit these distributions, finding a silver lining in the whole process. As part of the data collection, specific assumptions were recorded that related to key distribution parameters, such as pessimistic and optimistic levels. The documentation of these assumptions was communicated within management and turned out to be extremely valuable. "I didn't realize that you assessed the odds of signing up a targeted key customer to be as low as thirty percent" was a sample quote from a division manager. He then proceeded to try to do something about reducing and resolving the uncertainty. The process of thinking about, documenting, and communicating uncertainty became its own raison d'être.

As I fast-forward to the real estate finance problems that most commercial banks are faced with today, it is clear that assumptions made as to the viability of the proposed new commercial construction projects have often proved false. Further, today, valuations continue to be based on deterministic financial assumptions that will most likely also be substantially off the mark.

Appraisals are nothing other than simulations of assumptions, which are used to project property cash flows. For example, embedded in many valuation models are projections associated with tenants renewing their leases. These are usually stated in conditional probabilistic terms, such as a seventy percent renewal probability for a certain class of tenants, who if they renew will pay one level of rent, while if they do not renew, will require a reletting of space at a different rate. Yet, the normal appraisal approach is

simply to weight the rents, with an expected value result. It is clear that the application of Monte Carlo analysis of these and other variables could result in a probability distribution of value that might give lenders an improved basis for decision making.

The stumbling block, again, is the capability of appraisers or lending officers to construct probability distributions associated with these variables. However, the process of thinking through the factors creating these uncertainties could well lead to actions that could improve the investment returns.

I'm not aware of much progress in this area over the last twenty-five years, other than perhaps in the area of artificial intelligence and expert models. In glancing over last year's *Proceedings*, I'm struck by the number of papers covering tutorials, languages, computer-related systems, military applications, and manufacturing applications of scheduling and transportation networks. Yet, despite the boom in the financial markets from new financial instruments and the LBO craze, I see little application of financial models, other than a capital-budgeting model dealing with hog production—though the title alone could well sum up the financial feeding frenzy of Wall Street.

I would hope that more attention could be devoted to financial models, with the additional benefit of attracting some of Wall Street's rocket scientists to next year's conference.

Speaking of rocket scientists, I will conclude my remarks with a short story best titled "Simulation: Back to the Future." Several months ago, I was working with a colleague of mine who was not only a financial rocket scientist, but was in fact the son of a famous Nobel Prize winner in physics. We were trying to design a complicated security whose returns were to be tied to the performance of a real estate portfolio. We agreed that individual probability estimates of loss were required and even dreamed up some prototypical ones. But with the complicated relationships and the parameters of the security itself, we found that we were going to have some difficulty in estimating an overall probability distribution of returns for the security holder.

"No problem," he declared, "I'll work out the theoretical convolutions in a couple of days." Not wanting to be left in the dust, I weakly said, "Me, too." The ace I had was that I recalled having seen an ad for GPSS/PC and was able to track down Minuteman Software, who delivered a student version by Federal Express. Like riding a bicycle, it seems that you don't easily forget the basics, even after seventeen years; and I had my twenty-block model up and running quickly. Not quick enough though, because sure enough my colleague called me up and gave me the resulting distributions—albeit with what he acknowledged were "restrictive assumptions." One day later I duplicated his results—proving simulation is almost as good as theoretical results, particularly if you don't have a rocket scientist around to give you the theoretical stuff.

Naturally, the final triumph came when I eliminated the restrictive assumptions and let the model loose, and gained his acknowledgement that while it may not have been as elegant as his approach, it sure worked.

In summary, thanks for inviting me. I hope conferences such as these will devote additional effort to improve the quality of the data and the assumptions associated with financial models, and that they will continue to report on useful applications of simulations—that is, applications that work.

HAROLD G. HIXSON

REMINISCENCES OF THE FIRST CONFERENCE

First, let's explore the EXTERNAL EVENTS that influenced the scenario of the first of these twenty-five conferences. There are two primary benefits from a pursuit of historical truth: history is just plain interesting, and we may even learn something valuable from it. We revere the *Guinness Book of World Records* for both reasons. Each world record is interesting. And each demonstrated limit of skill and endurance gives potential champions a mark to surpass. Examples of each in the realm of simulation are:

- Wouldn't it be interesting if we knew who did the first publicized computer simulation? Of course, a report of the very first simulation had to be published for us to know about it. I have a candidate for that honor. If this is not really the earliest, then it must come close. According to the *Proceedings* of the IBM Computation Seminar held in August, 1951, as early as November 1949, W. W. Woodbury ran a Monte Carlo simulation using an IBM Card Programmed Electronic Calculator. And incidentally, who attended that seminar in 1951 whose names we might remember? I'll mention two—Herb Grosch, from the Technical Computing Bureau of IBM, and Richard Hamming, from Bell Labs.

- As for learning from simulation history, according to the program for this conference we are still testing random number generators empirically. After-the-fact analysis is too late—of course that's what we tell our customers and clients all the time. And we certainly should practice what we preach! As early as 1961, mathematical results from number theory were used to establish the period of a given generator on a specified computer. The advantages of this are to avoid generating long sequences of random numbers merely for input to a statistical analysis and even to avoid the statistical analysis. But we can still tailor the generator to produce the random-number-stream period that we need for a given application. This bit of ancient history is virtually unknown, because few simulationists are mathematicians of that caliber and because the original paper was not widely disseminated. By the way, here is a quote from the past on this: John von Neumann said: "Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin."

Now we come to the matter of historical perspective. My goal has been to be multilingual concerning simulation languages and match the appropriate language to a particular application. Each application needs a particular combination of features, and a language with that combination can be used. Along with features, we consider mode of operation. Most languages do not have both an interpretive mode and a compiled mode. This means that any language that is solely compiled is handicapped by time-consuming recompilations during model development. Conversely, a language that is solely available in the interpretive mode has its execution time increased by inefficiency after model completion and implementation.

With this open-mindedness, I've compared language features to stay current on the state of the simulation art and to make informed choices. In fact I presented a paper "Comparison of Four Simulation Languages" at a simulation workshop at the Moore School of the University of Pennsylvania in 1966. It turns out that the history of these Winter Simulation Conferences is closely bound to the chronology of our tools—the simulation languages.

I've also been interested in the genealogy of simulation languages—a combination of history with comparison and contrast of language features. There have been only three broad families of languages in the discrete event simulation world. With my humble apologies to Jean Sammet (the computer language historian):

IN THE BEGINNING THERE WERE FORTRAN, ALGOL, AND GPSS. If you also wish to give credit to Jay Forrester's DYNAMO, a continuous-change-with-feedback-world-view modeling language, that's fine too. That language was sometimes applied to the applications we ordinarily address, e.g., in 1962 Max Kennedy applied DYNAMO to the study of Air Force aircraft component logistics for his master's thesis under Professor Forrester at MIT. In addition to the traditional mainframe implementations, there was at least one portable implementation in FORTRAN and a mini-DYNAMO for the Apple II (1983). Likewise if you wish to give some credit to Ken Iverson and his (1962) A Programming Language (APL), that's fine too. Claude Boucher, of Quebec, Canada, developed APLS to use for simulation in the conversational mode, which was the same mode used in 1965 by Mal Jones' OPS-3 (MIT). However, neither APLS nor OPS-3 became prominent. Counting the borderline cases DYNAMO and APL, there are five progenitors of most of the languages we at the Winter Simulation Conference use.

Anyway, back to our main progenitors, we pay homage to FORTRAN (1956), which spawned CSL, ECSL, BASIC, FORSIM IV, GASP, SIMSCRIPT, SEAL, GERTS, SLAM, PROSIM, and ASPOL. Strictly speaking, neither FORSIM IV (MITRE Corporation) nor GASP are languages, but rather collections of simulation-oriented FORTRAN subroutines. The Control and Simulation Language (CSL) and its successor ECSL were developed in England. SIMSCRIPT I and I.5 were followed by SIMSCRIPT II (also from the RAND Corporation) and II.5 (in multiple incarnations) and some extensions. One of the earliest of these extensions was ECSS I, the Extendible Computer System Simulator, followed by ECSS II. Another extension, developed about the same time by NCR, was QUIKSIM, similar to GASP and GPSS, but written in NCR SIMSCRIPT and thus compatible with FORTRAN. SEAL (Simulation, Evaluation, and Analysis Language) was a SIMSCRIPT look-alike distributed as a Type III program for the 360/370 series by IBM. NSS was also developed by IBM with emphasis on list processing but never achieved success. SALSIM was another SIMSCRIPT look-alike implemented using FORTRAN by TRW. For job-shop applications, an early extension of SIMSCRIPT I from the RAND Corporation was Programming by Questionnaire. This was followed a few years later by Automated Assembly of Simulation Models (AUTASIM) based on a choice of GASP II or SIMSCRIPT II. Currently some applications packages exist, notably for communications, networks, and factories.

Tracing FORTRAN descendants again, we see that the General Activity Simulation Program (GASP) was developed by Phil Kiviat at U.S. Steel (1963) and was extended by Alan Pritsker (GASP II and IV). GERTS, SLAM, SLAM II, and TESS were also developed by Pritsker. GERTS (Graphical Evaluation and Review Technique Simulation) existed in multiple versions, some of which were GERTS IIIQ, IIIR, and IIIZ. Simulation Language for Alternative Modeling (SLAM II) is noted for giving the modeler a choice of three world views. The Extended Simulation System (TESS) provides data-management, library, and other user-interface and convenience features for SLAM II, MAP/I, or GPSS/H. A number of books on his languages have been written by Alan Pritsker. Production System Simulator (PROSIM) was developed by Greenlawn and Hottenstein in 1969, and at least three more versions followed through PROSIM78. A Simulation Process Oriented Language (ASPOL) was developed in the early 1970s by Myron MacDougal at Control Data Corporation primarily to simulate computer systems.

Continuing the FORTRAN lineage in another direction, we encounter a full-featured language from IBM—Programming Language I (PL/I). PL/I was given simulation capability during the early 1970s by GASP_PL/I and SIMPL/I. GASP_PL/I is a PL/I version of GASP IV with dynamic storage allocation. SIMPL/I is a set of SIMSCRIPT-like extensions. The SIMPL/I statements follow the syntax pattern of PL/I and allow insertion of PL/I statements in the SIMPL/I source program. IBM also introduced SIM/PL, a PL/I version of SIMULA.

Special-purpose extensions of FORTRAN included the System and Software Simulator (S3) and the System Analysis Machine (SAM) developed by Leo Cohen and used primarily by the Army to simulate computer systems.

ALGOL, while it lived, (ALGOL 60 through ALGOL 68) was almost totally ignored, except in Europe and at Burroughs Corporation; but in the long run it proved to be the most prolific, if you count the number of descendants and the number of users of those languages.

SIMULA was an early descendant of ALGOL but was rarely used anywhere but Europe. However, SIMULA was implemented on the UNIVAC 1107 by its originators in Norway and therefore was available for use in the United States. From SIMULA sprang Smalltalk, LOGO, EIFFEL, and Hierarchical Simulation Language (HSL). Along with the MAD language, probably the next oldest offspring from ALGOL was JOVIAL, used primarily by the military for command-and-control systems. Some vestigial remains from JOVIAL were used in developing Nelliac, Coral, and PL/I. Another early descendant of ALGOL was the Simulation Oriented Language (SOL). SOL was used in at least three versions, primarily by the Army for communications and electronics simulations. Structured BASIC (SBASIC) was developed and taught at Dartmouth. Paul Roth and Al Meyrhoff developed the Burroughs Operational System Simulator (BOSS) in extended ALGOL. Then of course, came the ALGOL-like PASCAL language followed by MODULA II, ADA, and ELLIE (Denmark). Inspired by the these nonsimulation, but nonetheless powerful languages, came the modern simulation language MODSIM II and its proprietary extensions. Last but certainly not least, with its user base dwarfing most other languages (except the unmentionables like COBOL and 4GLs) is ALGOL-influenced C. A trend has been the movement of major languages to an underlying base of C—notably SIMSCRIPT II.5 and AutoMod II to facilitate porting to different platforms. Recently, with the advent of latter-day object programming, not only is PASCAL extended to Object PASCAL, but C has been extended to Objective C and C++. Simulation extensions have been added to many general-purpose languages. For example, Smalltalk, PASCAL, MODULA II, ADA, C, and C++ all have simulation packages such as TURBO SIM (Turbo Pascal), SIMPACK, and CSIM. The Hierarchical Simulation Language (HSL) interpreter is written in C++. Then there is ROSS, the object-oriented simulation system (1986).

I used the term *latter-day object programming* because SIMSCRIPT I made the first use of objects (called entities), attributes, and sets of entities, etc. Now, latter-day object programming includes objects, instance variables, methods, class encapsulation, the constructor and destructor functions, late binding, polymorphism, and inheritance (all of which provide flexibility and maintainability). Of course, classes and encapsulation were pioneered by SIMULA.

Finally we cover the General Purpose System Simulator (GPSS), which was a simulation rather than a general-scientific-language progenitor, coming on the scene in 1961. It was the first of the process-oriented languages, and that was long before SIMSCRIPT II.5 reformed its orientation from event to process! The year before (1960) had seen the introduction, by IBM, of a job-shop simulator. Of course, most of the versions of GPSS, from I through VI, were interpretive. The exception to the GPSS interpreter rule is the GPSS/H compiler. A competitor of GPSS/H is GPSS/PC, comparable to GPSS V with differences. An early GPSS look-alike in this country was SIM-

PAC, developed by System Development Corporation (1962). SIMCOM was developed by General Electric in 1964. Some GPSS look-alikes have been developed in Europe. The first of these was the General Simulation Program (GSP II) developed by K. D. Tocher of United Steel Companies Limited in England in 1962. And there is GPSS-FORTRAN, version 3, developed by Dr. Bernd Schmidt of Germany, who has written many books about his language. Proprietary GPSS extensions in the form of even higher level languages for certain classes of applications are widely used. One early example in the GPSS style for a special purpose was the Computer System Simulator/360 developed by IBM. Other contemporary extensions are AutoMod II and the questionnaire approach of the Automatic Model Generators. To some extent the very modern SIMPLE I and SIMAN languages are like GPSS, although SIMAN features experimental frames.

What can we learn from all this simulation language history? It is obvious that language developers have been wildly prolific. We certainly have seen widely diverse exploratory efforts pushing in practically all directions. There has also been a lot of fusion of languages. For example, how do you classify GPSS-FORTRAN—a version of GPSS or an extension of FORTRAN, or simply a merger of both? So, during the next twenty-five years, will the major simulation languages stabilize and new developments taper off; or will we see the emergence of another three-dozen languages? Some of these language innovations were vital to the orderly evolution of simulation languages, but others seemed to be unnecessarily duplicative without much "value added." We do need a new simulation language especially designed for parallel simulation. Modest capability for this now exists with Concurrent C or ADA, but the powerful syntax to which we've grown accustomed in simulation and the ability to keep multiple processors busy is limited in these two languages.

By the way, for you trivia buffs, after a few years IBM changed the name of GPSS to General Purpose Simulation System! The list of alias names for GPSS as implemented by other computer manufacturers is long and includes GPS K, GESIM, GPDS, etc. The early name that preceded ALGOL (ALGOrithmic Language) was IAL (International Algorithmic Language). IAL has another claim to fame—it was preceded by the words "Our version of" to make the name OVIAL, which was immediately superseded by the term "Jules own version of" to create the name JOVIAL, in honor of the leader of that language's design team, Jules Schwartz. Also, PL/I was first referred to within IBM as FORTRAN 6 and later as NPL (New Programming Language). The Cornell List Processor (CLP) was used for simulation in 1963. ESP, the Elliot Simulator Package, was developed in 1964. Niklaus Wirth's experimental language EULER (1966) preceded his more illustrious PASCAL (1971). BLISS, a language without labels or GOTOs, was born during that era. The first econometrics model of the United States economy was constructed by J. Tinbergen in 1939. Systematic development of Monte Carlo methods began in 1944! Ah, nostalgia, isn't it great for transcending time and space?

We should at least mention some of the originators of the early simulation languages (the simulation language pioneers). Simulation "Oscars" should go to Harry Markowitz (SIMSCRIPT I and II), Phil Kiviat (SIMSCRIPT II), Geoffrey Gordon (General Purpose System Simulator), Ole-Johan Dahl and Kristen Nygaard (SIMULA), and Jay Forrester (DYNAMO).

Transitioning from languages to conferences, we observe that the history of the Winter Simulation Conference began long before the first conference. We who were active in simulation in the early 1960s knew that there was a lot of interest in attending conferences and exchanging information. IBM, as early as 1949, had held an annual IBM Computation Seminar, attended by the pioneers in the use of IBM computers. The IBM scientific computer users' group, SHARE, held meetings on a regular basis since 1955; and at some time in the early 1960s, a SHARE System Simulation Project was formed. A Symposium on Monte Carlo Methods was held in 1956, with its *Proceedings* published by John Wiley & Sons. IBM started to hold an annual Scientific Computing Symposium on Simulation Models and Gaming in 1959. A Workshop on Simulation Languages was held at Stamford University in 1964. That year an annual System Science Conference was begun at the Moore School of Electrical Engineering at the University of Pennsylvania in Philadelphia, which is famous as the birthplace in the early 1940s, of the Eckert and Mauchly digital computer, the ENIAC. As a matter of fact, Julian Reitman presented papers on simulation at those conferences. This series was followed by a Workshop on Simulation in 1966 on the initiative of Roger Sisson. Both Julian Reitman and I presented papers there.

In 1967 there was beginning to be a significant potential for a truly nationwide conference that would bring in people from all over the country (and all over the world). On a very small scale we had a prototype of the Winter Simulation Conference in the System Simulation Project of SHARE; and there was a counterpart of that project in Guide, which was a similar organization of business users of IBM computers. SHARE held full meetings every six months

with meetings of officers in between those. Most sessions were of high quality, with papers presented by leaders in the simulation field. However, there were only about six sessions at a meeting. Session attendance of one hundred people was unusual, with the average attendance less than fifty. The continuous system simulation languages were covered as well as the discrete event languages and their applications.

The unlikely event that I chaired the first Winter Simulation Conference is attributed to the facts that the SIMSCRIPT II language was brand-new and Phil Kiviat was the manager of the SHARE simulation project. As you might imagine, this leads to a short story: I was minding my own business at Wright-Patterson Air Force Base (March 1967). We were borrowing computer time across the base on an IBM 7090 computer because we had no computer of our own. Our venerable UNIVAC I computer had been retired by that time, and we had not yet accepted delivery of our new UNIVAC 1107. We had attempted to run GPSS I on an IBM 705, but for some reason on that character machine it was really unstable and did not always produce comprehensible output. So the binary machine—the IBM 709, 7090, 7094/7044 Direct Coupled System (as it was upgraded over the years)—produced our simulation results for a while. The computer center where we borrowed time was an early member of SHARE with installation code WC.

So everything was made to order for the entrance upon the scene of Phil Kiviat, who was at RAND Corporation in Santa Monica developing SIMSCRIPT II. He realized he was involved in too many extracurricular activities (and had to attend too many SHARE meetings, etc.), which were interfering with his work (of finishing SIMSCRIPT II), an untenable situation that could not be continued. Phil, working at the Air Force–workloaded think tank (RAND), came by periodically, advising our management on establishing a Simulation Center, of which I was one of the first members. He checked with the Director of the Simulation Center (Colonel Ken Swanson) who, incidentally, later on became the Keynote Speaker at the first conference. After getting a commitment of support from Colonel Swanson, Phil asked me to take his place as Manager of the System Simulation Project, which I accepted.

Arnold Ockene, of IBM, whom I got to know through my work with SHARE, was a prime mover in organizing the first Winter Simulation Conference (then called the Conference on Applications of Simulation Using GPSS). Julian Reitman and I got to know each other with his attendance and presentations at SHARE meetings. As a result of early discussions between the three of us, and mainly because I represented SHARE (one of the largest, most active, and most financially able of the sponsoring technical societies), I was asked to chair the first conference. The term *financially able* was the key to organizing a conference in the first place. Don't forget, we were attempting to stage a major conference with no financial resources! Our preliminary budget, courtesy of Ralph Layer, the ACM Conference Coordinator, submitted in July of 1967, conservatively projected conference attendance to be one hundred! We subsequently increased that estimate for planning purposes, but only up to 225. Some expenses had to be incurred prior to advance registration so a $700 advance was obtained from SHARE. With actual attendance of 401, we were able to repay it. Julie Reitman was my Program Chair and did a marvelous job. Arnie Ockene was Publicity Chair and really got the word around to everybody! Phil Kiviat was too busy as I described to take part in the first conference, but he was recruited for conference duty later.

As I said, the name of the first conference was "Conference on Applications of Simulation Using GPSS"; and it was cosponsored by SHARE and by groups in ACM and IEEE. We could announce at the conference, that besides IBM and UNIVAC, which already had GPSS on their computers, four other computer manufacturers had introduced versions of GPSS—Burroughs, Control Data, Honeywell, and RCA. And of course, Geoffrey Gordon himself spoke prior to the panel session on improving the language. By welcoming all the simulation languages to the second and successive conferences, we more than doubled our attendance.

Looking back over the *Proceedings* of the conferences, especially those of the early, formative years, the word *representativeness* stands out. We always want each model to be representative or similar to the real system in all important respects. However, that is not an easy task unless two conditions are met: that the real system is completely documented and that the complexities of the system are completely and accurately visualized in order to build a model and complete a successful simulation. For the following reasons, the quality of our simulation work should be better now than ever before:

1. Academic preparation of the simulationist and other participants is better.

2. Languages and computers are improved.

3. In many cases data for simulation comes automatically from computer systems.

4. Emphasis on quality makes data more accurate.

5. Management is more enlightened and receptive.

Are there any indications that simulation results are not better than ever? Is there anything that demonstrates that perhaps current simulation efforts are either done imperfectly or not done where needed? Unfortunately, the answer is yes. A relatively recent development, the Theory of Constraints (TOC), by Dr. Eli Goldblatt, concentrates on throughput, impediments to throughput, operating expense, and profit aspects of manufacturing plants and other facilities. Of course most facilities should have been subjected to simulation periodically during the last three decades. TOC is a manual empirical technique that is based on the implicit assumption that there is at least one and possibly many unknown constraints in each system. The existence of constraints in a system means they were either built in, or inadvertently added to, the system. However, if simulation had been applied successfully, there wouldn't be any unknown constraints in these systems. The success of TOC in the U.S. and around the world vividly warns us that many good simulation applications were either done superficially, temporarily, or perhaps not at all!

Without even looking at the old conference *Proceedings*, there's another word that comes to mind just from my firsthand knowledge of those who organize the Winter Simulation Conference each year. That word is *teamwork*. The fact that this annual conference has survived for twenty-five years shows that the team approach has been successful! It certainly has not been easy. One problem has been the conference competition. For a while in the 1970s, there was an annual Symposium on the Simulation of Computer Systems hosted by the National Bureau of Standards (first at Gaithersburg, Maryland and later at Boulder, Colorado). Although this occurred during our peak attendance years, it still must have had some impact even if slight. And the Annual Simulation Symposium in Florida has been in existence for the entire twenty-five year period. Its habit of siphoning off our conference attendance has been a continuous process and has been a carefully preserved attribute of that permanent entity. I understand that the current conference leadership is faced with similar competition from other quarters. I trust that ways and means will be found to assure the continuing success of the Winter Simulation Conference. It is living proof that teamwork between hardware and software vendors, the academic community, and users of simulation languages is accomplishing much of which we can be proud.

A requested postscript to these remarks addresses the parallel development of the Summer Computer Simulation Conference. I served as the General Chair of the first conference of that series also. The stage for that conference was also set at SHARE meetings, which as I mentioned, covered the full spectrum of simulation work, including the continuous system simulation languages. During the 1960s, a typical SHARE meeting would be attended by fifteen hundred individuals representing six hundred installations. Naturally, in 1968, there was much discussion concerning a nationwide conference that would serve the continuous system simulation community just as the already-established Winter Simulation Conference was serving the discrete-event-simulation community. Since I was a member of Simulation Councils, Incorporated (SCi), I was already attending regional simulation conferences, such as the ones held (normally on university campuses) in Chicago, Illinois and Toledo, Ohio.

Stanley Reed and Robert Brennan of IBM both attended SHARE meetings during that period, and discussions between the three of us progressed to the point of considering the individuals who were prominent practitioners of continuous system simulation and were members of SCi. It turned out that many of those interested, including Bob Brennan himself, were located in California, so that was proposed as the location for the first such conference. I was again approached as an experienced volunteer to be the General Chair, and I also favored a location where we would have sufficient manpower on site. So San Francisco's Sheraton Palace Hotel was finally selected as the conference location and June 30–July 1, 1969, were selected as meeting dates. We were careful to maintain a maximum separation in time from the winter conference to minimize the competition factor. We felt this was very important at the time, because future developments might include the combining of discrete event and continuous processes in the same simulation model. Because of this and other reasons, some people would want to attend both conferences each year. Some of the same people were involved in the planning for this conference, such as Ralph Layer (ACM) and Julian Reitman (IEEE). Thus, the Conference on Applications of Continuous System Simulation Languages was held under the sponsorship of ACM, IEEE, SHARE and SCi. Bob Brennan served as the Program Chair.

I quote from a letter written by David Brandin, the Conference Treasurer: "The Conference on Applications of CSSL was a smashing success—showing a net profit (after refunds of sponsor advances) of $388.28." David R. Miller, who was President of SCi at that

time, became the General Chair of the next (1970) Summer Computer Simulation Conference; and SCi (now The Society for Computer Simulation or SCS) has remained the steadfast sponsor of our (twin) sister conference as well as WSC ever since.

AUSTIN C. HOGGATT

A review of the draft of the technical program for WSC' 92 turns up the usual impressive list of application of simulation, and lots of work still going on to refine the representation of random process and in the use of statistical methods to measure the response of complex systems. Yet, there is a missing topic, namely chaos theory and its implications for the practice of simulation. The story that goes with this is as follows: when the Hoggatt-Balderston (1962) market simulator was under development, the host computer at UCLA was upgraded from a 709 to a 7090. I was in Hawaii at a conference reporting on some of our results, when a midnight call from Connie Hwang reported the disgusting news that the benchmark test on the new machine diverged after thirty periods of operation. That simulated market had over one hundred decision makers making thousands of decisions each period. I reported this at the conference and later traced it to a slight difference in hardware, namely, the floating divide in the 709 chopped and in the 7090 it rounded. One-half bit precision more, eventually, led to a reversal of a decision in the selection of a market trading-partner; and after that history was not the same. I did not fully understand the importance of this at the time and wrote that "economists ought not construct such ill-ordered models." Now Feigenbaum at Los Alamos Laboratories has shown the fundamental characteristics of systems of difference equations that precisely match the path of all simulations on digital computers. One of the noxious features is that history is very sensitive to initial conditions. If these models are reasonable representations of reality then prediction is a lost cause!

In economic modeling our difficulty arises with only two traders interacting via traditional demand and supply curves. I demonstrated this by a simple Lotus-based model driven by an old IBM PC. The substitution of `X * X` for `X exp 2` in one equation changed the attractor in the system (Hoggatt 1957). As practitioners we know that we should not, yet we continue to respond to managers in business and government who demand point predictions in an economy that is at least as sensitive to initial conditions as the simple models of chaos theory. Too often, practitioners respond to these demands only to be confounded when models and reality diverge. Recall President Reagan's lament that he had been fooled by an economist who preached supply-side economics. In fact, our models dwell in the region of their expected response surfaces only a small fraction of the time. This was shown in my thesis with studies of maximum likelihood estimators of single linear stochastic difference equations in which the maximum likelihood estimators of the individual coefficients were shown, not only to be hugely biased (a fact already known from statistical theory) but were also found to be unbounded for some regions of the parameter space (Hoggatt 1957).

Now for the moral of the story. We ought to focus on the frequency response of our models (Hoggatt 1989), characterize regions of instability, and warn managers to stay away from them, unless they are in situations with negative expectations where variance is the only hope. (Note the experience of the savings-and-loan-industry debacle.)

Of course, it is easy for me to give this advice—I am retired! Where does a salaried practitioner of simulation go for help in explaining to unreasonable managers that point predictions are the last thing that they ought to demand? May I suggest that we refer to some of the sponsoring societies, TIMS, ORSA, ..., and ask them to provide the professional arguments on which to base a vulgar appeal to authority.

All the best.

REFERENCES

Hoggatt, A. C. 1957. Simulation of the firm. IBM Research Monograph No. 6, IBM Corporation, Poughkeepsie, New York.

Hoggatt, A. C. 1959. An experimental business game. *Behavioral Science* 4(3): 192–203.

Hoggatt, A. C. 1989. Simulation, graphics, and data envelopment applied to the study of the stability of a model of the labor market. *Proceedings of the 3d World Conference on Mathematics at the Service of Man.*

Hoggatt, A. C., and F. E. Balderston. 1962. The simulation of market process. Research Memorandum, Institute of Business and Economic Research, University of California, Berkeley, California.

Hoggatt, A. C., K. A. Dawson, and G. J. Visser. 1990. Supercomputing and personal computing for teams of workstations. In *Scientific Excellence in Supercomputing: The IBM 1990 Contest Prize Papers*, ed. K. R. Billingsley, H. U. Brown III, and E. Derohanes, Vol. 2, 805–818.

Hoggatt, A. C., J. Escherick, and J. Wheeler. 1969. A laboratory to facilitate computer-controlled behav-

ioral experiments. *Administrative Science Quarterly* 14(2): 202–207.

Hoggatt, A. C., O. J. M. Smith, P. Smith, D. Northall, and S. Greenberg. 1991. Energy efficiency in Krakovian apartment buildings: An engineering and economic overview. Technical Report, U.S. Environmental Protection Agency, Office of Policy Planning and Evaluation, Climate Stabilization Branch, International Energy Program.

Hoggatt, A. C., and B. Wasik. 1992. Economic experiments to improve housing amenities and raise the energy efficiency of apartment dwelling in Eastern Europe. Technical Report, Cracow Academy of Economics, Cracow, Poland.

PHILIP J. KIVIAT

I became involved with, and active in organizing, the early Winter Simulation Conferences initially in order to meet people working in simulation, to learn what they were doing, and to introduce them to the work we were bringing to completion on SIMSCRIPT II at the RAND Corporation. Later on, after Arnie Ockene and I founded Simulation Associates to sell SIMSCRIPT II Plus and SIMSCRIPT and GPSS training and consulting, our interest in the conference and its success took on a more financial aspect.

I believe that people attended the early conference for, at least, five major reasons:

1. **To get an introduction to simulation concepts and methods.**

   For those of us who were early practitioners, this was an opportunity to share our enthusiasm, strut our stuff, and evangelize about a subject we felt passionate about. For many attendees, this was their first glimpse of what simulation was really about.

2. **To meet other developers of new simulation languages and concepts.**

   The late 1960s and early 1970s were years of innovation and development, and the hours spent sharing ideas on simulation-model world view, simulation-language structure, and programming approach were some of the most exciting I ever had.

3. **To hear about simulation modeling successes and failures.**

   Getting practical insights from both successful and unsuccessful practitioners gave language developers very important feedback, and language users gained a lot by sharing techniques and experiences.

4. **To learn how to use simulation models correctly and intelligently.**

   George Fishman was an early pioneer in the search for understanding how to use models to make reliable business decisions. There seems to be a never-ending search for techniques to represent the statistical processes we model and to most efficiently derive reliable estimates of the outcomes our models are designed to produce.

5. **To meet people with a common interest in simulation.**

   Networking is the popular word for this. It was important to all of us early on. It is no less important today.

All these reasons are still valid, although the emphasis is perhaps different since the field has matured in twenty-five years. A Winter Simulation Conference was needed in 1967 to get the field started; it is needed now to educate and energize current practitioners and proselytize new entries to the field; and it will be needed tomorrow to provide a continuing forum for new ideas, to provide exposure for successful applications, especially in new areas, and to provide a networking infrastructure for all of us and for our successors.

MICHAEL F. MORRIS

PREDICTING BREAKTHROUGHS IN COMPUTER TECHNOLOGY

Introduction

The convening of the twenty-fifth Winter Simulation Conference is a nostalgic event for me so I've written a light, reminiscent article about an experience I had to show that useful results can even come from elementary application of the simplest kind of modeling—linear projections of historical data. The project was done mainly in spare moments because I didn't think I could get real answers so I didn't begin to take it seriously until I had collected and studied a lot of almost random historical data about computers.

The project took place in 1989, from February to September, and I have not written of it in a published journal or proceedings until now. By the time I could give answers to the question, I had become so fascinated by the topic that I have continued to this day to add "random" historical information to my database. I hope you find the topic as interesting as I have.

The Question(s)

It is easy to plan work several years in the future when there is reasonable understanding of what the tools then available will be able to do and at what cost. Most tools haven't changed much over the years. Consider the hand tools that we all have around our homes: electric or pneumatic motors have been added to some, but a user who understood the application of a hammer, saw, screwdriver, etc. at the turn of the twentieth century would have no problem applying the most modern versions of such tools nearly a hundred years later. The centenarian carpenter would surely find the latest hand tools to be more user-friendly and less demanding of his muscle power, but their technology is unchanged and their prices have changed in line with the general economy.

The technology of computer systems, the tools of our trade, has undergone remarkable change in less than fifty years. The cost per unit of computer power and storage has dropped while most other prices increased. Computer-technology breakthroughs happen so rapidly that it seems impossible to predict what our tools will be or will cost even two or three years in the future.

Early in 1989 a client asked me how much space it would take and how much money it would cost in 1995 for a computer system approximately equal in power and storage capacity to the then-most-powerful IBM mainframe—the IBM 3090/600S. This was important for planning the NASA space station: the client needed to know how much money to budget for this system and how much space to allocate for it.

Although this may seem an esoteric question to ask, I also got almost the same question from more down-to-earth clients. These were usually in the form: "Our processing load will double in two (or three, or four, ...) years; how much will it then cost for a processor to handle this increase?" Or, "We are planning to consolidate all of our computer centers in two years and move to a 'lights-out' environment. We'll need three times as much processor power and five times as much storage as we have now. How much space do we need for this and what will the processors and storage cost?" And so forth. These are routine questions for computer-center managers to ask. I wondered why it should be impossible to predict reasonable answers.

Data Collection

Without much confidence of ever getting good answers but as a believer in the simulation approach, I began to collect data. I gathered everything I could find that might describe how processor power and size, storage density, and costs varied over time. An early compromise I had to accept was to use MIPS (millions of instructions per second; also meaningless indicator of processor speed) as the metric for processor power. This because, bad as MIPS may be, it was the only performance measure I could find or estimate myself for the full range of computer systems over a long period.

As with most modeling projects, far more data were collected than were needed. The resulting "database" is available from me upon request. I hope to use this database for an article or book on the history of computers one day. A few items from this list suggested an approach.

In 1955 the IBM SAGE (Semi-Automatic Ground Environment—a computer supporting an early-warning radar network) was the state-of-the-art processor. SAGE is physically the largest computer ever built. It contained 58,000 vacuum tubes, weighed 113 tons, consumed more that one megawatt of electrical power (enough to power a modest-sized town), and cost $2,000,000 plus the cost of a four-story building to house it. SAGE ran at about 2,000 instructions per second (0.002 MIPS). The last SAGE went out of service in 1981—a twenty-six-year system life.

In 1989 Intel announced the I-960. It was then the state-of-the-art processor. The I-960 contains 600,000 circuits (a circuit typically has much more capability than a vacuum tube), it weighs a fraction of an ounce, draws less than one milliwatt, and uses about 0.22 square inches of real estate. (As for the SAGE, the customer furnishes the container.) The I-960 runs at about 66 million instructions per second—66 MIPS. In lots of one thousand, its initial cost was $325 apiece (see Table 1).

Storage comparisons were surprisingly similar. In 1951 magnetic drums had an areal density of 1,000 bits per square inch. In 1989, thirty-eight years later, laser tape ("digital paper") reached 400,000,000 bits per square inch. That's about an order of magnitude every seven years. And IBM's 8-mm helical-scan magnetic tape reached one gigabit per square inch in 1992 (see Table 2).

Definition: Practical Breakthrough

These comparisons showed that several technological breakthroughs took place, but I couldn't begin to identify and classify the myriad of specific discoveries and inventions that allowed the progress from the SAGE to the I-960 or from magnetic drum to digital paper. I noted that the products are orders of magnitude apart in every way they are compared, so I made up a definition for what I called *practical breakthroughs.*

Table 1: Comparison of Processors

Characteristic	Processor		
	SAGE	I-960	I-8088
MIPS	0.002	66	0.12
Circuits	58,000 vac. tubes	600,000 VLSIC	64,000 LSIC
Power	> 1 MW	< 1 mW	
Weight	113 tons	< 0.25 oz.	
Size	4-story bldg.	< 0.25 cu. in.	
Cost	$2,000,000	$325	$10
System life	1955–1981	1989–1995?	1980–1988

Table 2: Comparison of Storage Devices

Year	Device	Areal Density bits/sq. in.
1951	Magnetic drum	1,000
1989	Digital paper	400,000,000
1992	8-mm helical-scan magnetic tape	1,000,000,000

Table 3: Average Power and $/MIPS for Mainframe Processors*

	Small		Medium		Large		Very Large	
Year	MIPS	$/MIPS	MIPS	$/MIPS	MIPS	$/MIPS	MIPS	$/MIPS
1980	0.32	188,378	0.9	219,290	7	373,580	NA	NA
1981	NA†	NA	1.1	203,692	10	247,573	NA	NA
1982	0.40	164,845	0.7	207,659	14	259,690	NA	NA
1983	0.42	160,202	2.1	171,532	17	290,514	NA	NA
1984	1.0	65,000	4.1	160,390	23	235,032	NA	NA
1985	0.93	45,651	4.1	150,644	34	172,336	NA	NA
1986	2.9	21,028	5.9	111,739	33	173,141	74	171,486
1987	6.5	8,158	13.0	65,984	45	155,434	76	152,813
1988	5.1	5,209	14.0	45,813	47	122,272	89	131,720
1989	14.0	3,561	28.0	20,431	75	109,760	116	113,720
1990	21.0	2,183	51.0	3,686	80	98,018	148	111,486
1991	50.0	902	93.0	1,667	117	66,212	187	100,422
1992	NA	NA	NA	NA	NA	NA	313	94,560

*Based on data available for announced multiuser computer systems, or mainframes, categorized by total initial price of a basic configuration: small, $10 thousand–$100 thousand; medium, $100 thousand–$1 million; large, $1 million–$10 million; very large, over $10 million. Single-user computer systems (PCs and workstations) ranged from an average of 0.4 MIPS and $15,063/MIPS in 1985, to 16.2 MIPS and $479/MIPS in 1991. These data are based only on those releases received by this author and tend to overemphasize faster systems that are more highly publicized. Most notably, the comment on single-user computers does not include the large number of off-brand PCs which would probably make the MIPS and $/MIPS lower than stated here.

†NA means that not enough information is available.

A *practical breakthrough* takes place when, beginning at any time with known MIPS for a current processor, areal density for current storage, and prices in $/MIPS and $/megabyte, later systems have ten times as much power expressed in MIPS, and ten times more bits per square inch of storage, and when the later costs are one-tenth as much as the original systems. In other words, when speed and storage increase by an order of magnitude and price decreases by an order of magnitude, there has been a practical breakthrough.

(I acknowledge that many important variables are ignored by this definition. It does not include important factors like channel speed, cache memory, scalable processors, or software advances, to name just a few. I hope someone picks up on this and refines it into a comprehensive computer system model. I'm too tired.)

Analysis

Looking again at the processor trends, we see that going from 2,000 to 66,000,000 instructions per second is more than four orders of magnitude (3.3×10^4) over thirty-six years. That's less than nine years per order of magnitude for processors. This comparison may be too far-fetched to be convincing. Let's just look at the past decade. The Intel 8088 was introduced in 1980. It was rated at 120 KIPS or 0.12 MIPS (Table 1). The I-960 represents an order-of-magnitude improvement each 3.6 years over the 8088. That's three to four years in the last decade versus nine years over the last three decades. Processor breakthroughs are happening much faster now. (Table 3 shows data across averages for a range of mainframes, but the information available to me at the time was too sparse to use.)

Likewise, our comparison of magnetic drum versus digital paper may be too extreme an example to believe. In the past decade, Winchester drives have had density improvements of an order of magnitude about every five years (see Table 4). This is not quite as dramatic as processors, but close. (And now laser and RAID technologies are replacing the Winchester approach.) Cost for both storage and processors has been decreasing by an order of magnitude about every five years as well for the past decade. (And I haven't even touched on software advances.)

Table 5, "Weight and Size of Various Media to Store One Terabyte," is included here without discussion to suggest the kind of information used to arrive at the size of future systems. Processor size was ignored: it is now insignificant to most space considerations.

The *S*-Curve Again

The well-known *S*-curve of progress or technology usage versus time is clearly in effect here. Without supplying the data, it was apparent that the initial relatively flat progress in the factors of interest spanned about thirty years from the 1940s into the 1970s. The progress-rate "knee" turning the *S*-curve upwards took place early in the 1970s. Several factors contributed to this. Among them were: large-scale integration providing several circuits on a single chip began the era of rapid miniaturization in 1969; DRAM was developed in 1970; monolithic circuits and microcomputers became practical realities in 1971; Winchester disks were introduced in 1973; and more. The progress rate became dramatically steep beginning early in the 1980s and it continues to be at least as steep today. The 8088-versus-I-960 and Winchester comparisons demonstrate this steep growth in the rate of progress during the 1980s.

The Answer(s)

Putting all this together, I estimated that a practical breakthrough was occurring at least once each five years, probably closer to once each three years. I also concluded that, if the rate continued to change as it had in the last few years of the 1980s, practical breakthroughs would occur about once each two years by the year 2000.

I made what I thought was a conservative estimate that the space-station project should expect to spend about $100,000 and allocate about four cubic feet for the equivalent of an IBM 3090/600S in 1995. Events of the three years since I made this estimate have already proven me wrong. This price and almost this size are available today, in 1992. Two examples are the 116 MIPS AViiON 8000 mainframe-in-a-pizza-box from Data General and the Infinity 90 series from Encore (however many MIPS you want). The plans for systems based on such architectures as announced for Digital Equipment Corporation's Alpha processor and rapid progress in adoption of the SuperbusPlus as an industry standard suggest that we "ain't seen nothin' yet!"

Playing with my original data produced the surprising prediction that the power of a 1989 Cray YMP with more than a terabyte of storage should cost less than $10,000 by the period 2005–2010. And, except for the keyboard and display (if such are then necessary), it should fit in your shirt pocket. A reported 1990 industry study used detailed analyses of specific computer system components and individual engineering estimates of how these would change and improve to come to a comparable conclusion: Their

Table 4: Comparison of Capacity and Cost for Winchester Drives

Year	Capacity MB	Cost $	Cost/Capacity $/MB
1980	5	$1500	$300.00
1984	10	1995	199.00
1985	20	2195	110.00
1988	80	550	6.90
1989	330	1450	4.40

Table 5: Weight and Size of Various Media to Store One Terabyte

Medium	Pounds	# Units	Square Feet
Digital paper tape	2	1	$\ll 1$
VHS T-120 Cassettes	60	100	< 1
CD-ROM	389	1,852	4
IBM 3490E tape cartrg.	1,139	2,778	60
IBM 3480 tape cartrg.	2,278	5,556	120
6250 bpi mag. tape	12,852	5,556	270
Microfiche	16,204	1,851,852	370
$5\frac{1}{2}$-in. floppy	30,245	833,333	1800
1600 bpi mag. tape	51,389	24,074	1380
$8\frac{1}{2}$-by-11-in. paper	5,250,000	500,000,000	Indeterminate

YMP-in-a-shirt-pocket would arrive before 2012 and cost less than $1,000. (The exception of keyboards and displays is because the sizes of these devices are no longer subject to technological progress. Their size limits are consequences of human factors. Unless our hands and fingers get much smaller and our vision much better, there is no point in downsizing this class of devices.)

A Few Quotations

Finally, I'm not surprised that my projection was too pessimistic. Pessimism has been the rule when attempting to foresee the future of computers. Here are a few quotes that I ran across when collecting information for my "database" to show that I'm in good company. Notice how their tone changes from when computers are in their initial flat-rate-of-change period to when they are in the rapid-rate-of-change part of the *S*-curve.

> I think there is a world market for about five computers.
>
> —Thomas J. Watson, Chairman, IBM Corporation (1943)

> Nine SSECs will handle all U.S. computing tasks for decades.
>
> —IBM market study (1949)

> The computer has just about had it.
>
> —Wall Street analyst (1971)

> There is no reason for any individual to have a computer in their home.
>
> —Ken Olsen, President, Digital Equipment Corporation (1977)

> The computer is "Man of the Year" for 1983.
>
> —*Time* (1984)

> If the automobile business had developed like the computer business, a Rolls-Royce would now cost $2.75 and run three million miles on one gallon of gas.
>
> —*Time* (1988)

> And several of these Rolls-Royces could be parked on the head of a pin.
>
> —Michael F. Morris (1992)

> IBM Corporation and several prominent scientists say optics may be good for some things ... but doing complex digital computations simply isn't among them.
>
> —*Wall Street Journal* (January 30, 1990)

(The pessimism voiced in the last quote might signal the beginning of an entirely new *S*-curve for optical-processor technology.)

ARNOLD OCKENE

Some aspects of the first WSC in New York remain clearly etched in memory; others are but dimly perceived through the haze of twenty-five years. In 1967 I was responsible for the support and marketing of IBM's General Purpose Simulation System (GPSS). It would be two more years until IBM's watershed "unbundling" announcement, so that no revenue could be attributed directly to a software product such as GPSS. Application software served the role of adding value to the computers on which it ran, as well as consuming machine resources and thereby helping to justify additional hardware purchases in the future.

Among the many customers with whom I had frequent contact in those early years was Julian Reitman. Julian was both a prolific model-builder and a demanding advocate for ever more function in GPSS, which had undergone rapid development in the mid-1960s. Julian's group in the Norden Division of United Technologies was among the most creative in "pushing the envelope of possibilities" for discrete simulation, and many of the improvements that appeared in a series of GPSS releases over several years were the result of the continual stream of requirements emanating from groups such as his. Both Julian and I had an intense interest in applications, and at some point in our frequent conversations the topic of a conference devoted to applications of GPSS arose.

We had little doubt that the time was appropriate for such a conference. Two earlier meetings on simulation languages, one at Stamford University in 1964 and the other at the University of Pennsylvania in 1966, indicated a need to change focus from a concern with the languages themselves to the application of those languages. Since both Julian and I,

he as a user and I as a focal point for requirements and marketing in IBM, were concerned with GPSS, the first meeting was limited to applications of this language. Sponsorship was secured from ACM, IEEE and SHARE; and Harold Hixson, an active GPSS user at Wright-Patterson Air Force Base, agreed to take on the duties of General Chair. Julian became Program Chair, and he put together a group of twenty speakers covering a broad spectrum of applications. We settled on two days in November of 1967, in New York.

Our expectations for this first applications conference were modest, and we booked a room at the New York Hilton capable of holding two hundred people comfortably. We wound up with 401 registrants, reaching a point where we had to discourage others from attending since we were unable to move the meeting to a room large enough for even the 401; I can still see people standing along both sides and the rear of an uncomfortably warm meeting room. The applications focus had apparently hit a responsive chord!

The success of this first meeting quickly led to plans for a Second Conference on Applications of Simulation in December 1968. Julian was General Chair and I became Program Chair. The program expanded to twenty-two sessions spread over three days, with eighty papers presented in areas ranging from financial models to the production floor to human behavior. Sponsorship expanded to include Simulation Councils, Incorporated (SCi) in addition to the earlier sponsors. Optimistically planning for attendance in the range 600–700, we were once again surprised by the turnout of 856 registrants.

After the 1968 meeting the conference took on a life of its own. The 1969 meeting in Los Angeles added the American Institute of Industrial Engineers (AIIE) and The Institute of Management Sciences (TIMS) to its list of sponsors. Both the 1970 and 1971 meetings were held in New York; the 1970 meeting, following the pattern of previous meetings, was called the Fourth Conference on Applications of Simulation. The first meeting to carry the name Winter Simulation Conference was the 1971 meeting. Attendance peaked with the 1971 meeting, at around 1,200.

I'll leave it to others to describe the evolution that followed, since my career moved away from simulation in 1971 and I stopped attending the annual conferences. It is, of course, gratifying to see that the meeting has had sustained value, as evidenced by this twenty-fifth anniversary session. Here's looking forward to the next twenty-five!

JULIAN REITMAN

HOW THE HARDWARE AND SOFTWARE WORLD OF 1967 CONSPIRED (INTERACTED?) TO PRODUCE THE FIRST IN THE SERIES OF WINTER SIMULATION CONFERENCES

Today, as we look back over twenty-five years of a strong and continuous conference devoted to one major subject, it is useful to go back to try and retrieve the environment that spawned this endeavor. Discrete-event simulation was not then and is not now part of the mainstream of either computer theory or applications. To a large degree, it developed as a counterelement in the emerging computer culture—a tool for support in an area that aimed toward the practical rather than the theoretical, and toward the complex rather than the simple. It is my purpose in this very personal history to go back to those early days and point out the ways in which the emerging capabilities of computers were directed toward discrete-event simulation with minimal, yet crucial organizational support.

The approach I have selected is to present initially some of my personal experiences that demonstrated the need for discrete-event simulation and how, in each case, I handled the situation before these tools evolved. And then, I describe the technical environment that existed forty years ago when the first steps leading to the initial specific conference directed toward the application of simulation to practical problems. In the process, I hope to show how engineers' capabilities to solve systems-design problems evolved and provided a foundation for future developments. The method I used to retrieve these memories is not scientific or historically researched, just what came to mind over the days devoted to putting these thoughts onto the monitor and looking over the program of the 1967 conference and the residue of what were once extensive files.

My introduction to the need for simulation probably goes back to my service in the U.S. Army Air Force in World War II. From that experience, I became convinced that electronics could address problems that were otherwise unsolvable—the optimism of youth—and that to solve these problems a systems view was needed.

So in a sense, the start of recognition of a need for discrete-event simulation goes back to World War II and the development of complex military electronic systems. Wartime radar systems provided a prime example. These systems had the requirement to process

data from multiple sources in real time and present information in a manner that helped operators make decisions. With the data-manipulating tools available at the time, the systems were structured to schedule rather than process the data on demand.

During the war a considerable body of theory emerged from the experiences of designing, building, testing, and operating these systems. With the computational tools available at the time, mathematical approaches were used to produce approximate answers even if the results were different from the real world. The basic theoretical solutions were modified by practical experience, but the feeling was that a mathematical approach would get close enough to the desired solution. The postwar emphasis was on a theoretical approach rather than a focus on experimental trial-and-error efforts. Trial and error was considered less convergent to a solution and wasteful of resources. Of course there are those who remember the development of radar differently. But, the point I am making is that shortly after the war many volumes, *The Radiation Series*, were published as a unified set and these became influential. This series of books emphasized the role of theory. Practical engineering was considered less up-to-date.

My experiences during World War II brought some of these differences between theory and practice into conflict. I had started my engineering education at City College of New York during the war and had been drafted into the USAAF, where I eventually went to radar school. After the war, I returned to CCNY, an engineering school still devoted to electric power, where electronics was mostly ignored, except for a little on radio. Complex systems that used electronics for control or communication were not part of the curriculum. There was a conflict between practical experiences in electronics and electronic systems gained in the service, and courses devoted only to elements of power systems. There was nothing in the curriculum about designing electronic control or complex systems.

With that background, I was lucky that my first job in the commercial world was in acoustics, a theoretical field; but I was assigned to design and supervise electroacoustic test equipment. Test equipments are not individual items, but part of a test system where failure of one part can stop the entire production process. This was an ideal location for a young engineer to be introduced to Murphy's Law, "Anything that can go wrong, will, and at the most inopportune time." In those times there were manufacturing plants located in Manhattan. This enabled us to use our break for lunch to tour Canal Street and see if any war-surplus electronics could be converted to our test purposes. Surprisingly, that frequently was the case. One side effect was that when that gear failed, we sometimes had no idea how to get it back in service. This certainly introduced the need for a systems concept when we constructed new test systems.

An additional benefit of working in acoustics was being introduced to the concept of dynamic analogies. One of the approaches to the design of electroacoustic devices was to employ the components of electrical circuits to predict the performance of the mechanical and acoustic aspects of the system. This is a form of analog simulation where the frequency response of the electronic circuit would, according to the theory, be similar to that of the acoustic circuit as theoretically represented. It was easy to change the value of a resistor or capacitor and note the change in frequency response, something that otherwise required considerable time in the machine shop to fabricate a number of new parts and then perform the always-difficult task of acoustic testing.

This was also the time when general-purpose analog computers became available. In my next job, on the basis of my experience with acoustic systems, I had the responsibility of using an analog computer to simulate how much aid a human operator should have to arrive most quickly at the desired end, dynamically tracking an object. It was a most educational experience, as at the time I was taking an advanced degree at New York University, which still had an engineering school. One particular area of concentration was servomechanism theory. The theory predicted that the system would behave in a particular manner; fortunately, I had access to a working system, and it did not behave according to the theory. After delving into the details of the theory, I finally used a scope to see what was going on in the analog computer circuits and found out to my great surprise that the system being simulated was operating not linearly according to the theory, but totally nonlinearly in a saturation mode. The operational amplifiers were either driven to saturation in one direction or the other—you might call it digital control. As you might gather, this was not the anticipated result; the theorists were still not ready to recognize reality and develop a new theory, which they eventually did long after my association with the problem had ended.

For me the lesson was clear. Current theory was useful, if you understood when and where to apply it and what the limitations were. That message was not part of the engineering culture of the decade of the 1950s. That culture emphasized theory over practice. To too-large an extent, that culture has continued to exist.

This perspective was reinforced while I was at

the Teleregister Corporation in the mid and late 1950s. We were designing airline-passenger-space-availability systems for a number of carriers. The Teleregister reservations system used telephone lines to connect remote agent sets to the central computer. Initially dedicated leased telephone lines were used for data transmission. Later, negotiations were successfully completed with the Bell System to allow dial-up lines to be connected to the Teleregister computer. This commercial application of an interconnection between telephone lines and computers was operating in the 1950s, in real time, in full sight of the general public, and long before military interconnection ties were established. It provides an example of computer historians ignoring the real world when presenting their story. The Teleregister operation in the 1950s was of large scale, as every major airline except Eastern used these systems. This is an area of computer history that is totally ignored. Computer historians prefer to dwell on military systems as the military bureaucracy has left a well-marked paper trail. Unfortunately, commercial computer systems were not documented; they just did (in some cases) what they were designed to do, even when that meant tying into Ma Bell's telephone lines.

The Teleregister digital-computer-based, real-time systems were providing airline-reservation, ticket-office, and airport agents with the information to book a seat for the customer's desired time, or to try and change the customer to use an alternative available flight. In that prejet time, aircraft were small, had a limited number of seats, and flew in short hops with many intermediate stops from origin to destination. Seat-availability information had to be provided to a site remote from the computer, twenty-four hours a day, and up to a rate that was within what we, the system designers, had predicted as peak traffic. Our fear was that the system would fail due to inability to handle peak traffic. The airline agent would end up waiting, holding the customer on the telephone; and the quality of service would decline at exactly the time the carrier was most interested in service, the peak period.

In the theoretically oriented manner of the times, we approached the design problem by investigating the available body of queuing theory. Queuing theory was the established tool; the system was expected to fit into the structural restrictions inherent in the assumed behavior of the public. There had been a considerable body of effort in the early part of the twentieth century, especially by Erlang, to predict the behavior of telephone systems, also under peak conditions. The mathematical approaches to traffic prediction worked under normal conditions for the telephone systems. However, it did not work to predict telephone service during snowstorms when everybody was home and calling their neighbors; but we were unaware then of that class of failure. Instead, we systems engineers performed those theoretical queuing calculations and checked them against operational data gathered from our installed computer-reservations systems.

Unfortunately, those data gathered from our field installations did not conform to the queuing-theory predictions. A basic assumption of queuing theory, as applied to telephone systems, is that every call is independent, not directly related to other calls. Our studies indicated quite the contrary: when seat availability was limited—a busy period—the reservations agent would introduce numerous transactions to search for an alternative space for the potential passenger. These transactions were not independent; instead, they were bunched in a short time-interval and were most likely during the period when passenger reservations traffic was heaviest. A few of these transactions, if they occurred in a short interval, could tie up the entire system with a peak far beyond our predictions.

We at Teleregister were faced with the problem of how to design and predict system performance in peak periods for a communications/computer interface when the available theory failed to provide adequate results. As we became aware of the weakness of theoretical queuing analysis for our computer-controlled real-time systems, we proceeded to set up and perform a manual simulation of the traffic under extreme conditions. I remember the exercise very well. For an entire week, the table of random numbers was manually consulted, transactions were generated, and a printing calculator was used to perform the mathematical operations as we wanted to preserve an audit trail. The result of that one week's effort was one point on the curve. To obtain a desired set of points would require the training and staffing of an entire group of competent desk-calculator operators. That unreasonable level of effort meant that to obtain adequate information to design systems, we needed a better approach. The Teleregister management understood the problem and funded an effort to develop a program for computer simulation. That effort was beyond the state of the art in the late 1950s. We failed as we were unable to produce results quickly, in a cost effective manner, and without requiring extremely skilled people. There the matter rested until I next encountered a need for simulation in 1961 at Norden.

While I was struggling with queuing problems at Teleregister, telephone-system designers had come to

the conclusion that the complexity of telephone systems ruled out the use of computer simulation with the resources available at that time. There was little incentive to proceed with an alternative to mathematical analysis. Geoffrey Gordon, then at Bell Labs, was out of step with the prevailing attitudes there, so he left and joined IBM. He brought his ideas for simulation with him—ideas based on his experience with analog computers, where the engineer solved the problem without obtaining help from programmers. At IBM the climate was more favorable and he quickly completed his first discrete event simulation system, GPSS I, in 1961.

The United Aircraft Corporation, which included Norden, was at the time one of the largest scientific computer customers of IBM. When IBM explained to us that they were using an internal simulator to evaluate computer-component performance, we suggested that the simulator be made available to us. IBM changed policy; and at the Eastern Joint Computer Conference in Washington in December 1961, Geoffrey Gordon presented his paper and GPSS was released to IBM customers. At the time, software was bundled with the purchase of hardware. We obtained a deck of source cards, a four-day training course, and some manuals. With that extensive preparation we were able to experiment with GPSS I.

The environment in which we experimented with GPSS I was that of a large mainframe. The simulation problem was submitted as a deck of punched cards with job control that called for the mounting and reading of the GPSS I program from magnetic tape. The output from running the program went to an output magnetic tape and was printed in reverse order, large printouts last. This operational environment was far from user-friendly, especially when a trivial keypunch or logic error resulted in a GPSS running error and a very large printout, usually of all the possible transactions. In practical terms, we were able to try one GPSS run per day, submitting the job after getting last night's printout, discovering the trivial error, correcting that error, and submitting the job again. After many days the process would converge. Simulation was still far from a quick-and-easy tool; but once a model was debugged, there was a reasonable probability that a series of parametric runs would work.

The attitude in United Aircraft Corporation was supportive; and a series of simulation models was built to determine the utility of this new tool, especially after improved GPSS II and III became available and we learned ways to limit the size of our printouts. One model for Sikorsky compared the repair times for different configurations of engine and reduction-gear assemblies. Another, with inputs from Sikorsky, the Research Laboratories, and Norden, modeled the use of helicopters with dipping sonars to protect a convoy against a coordinated attack by several submarines. This model was reported on at the 1967 conference, but without the interesting results. If data from helicopter sources were used, the helicopter was successful; however, when data from submarine sources were used, they were successful. We were never able to obtain agreement on input data. The irony at that time was the existence of a GPSS model evaluating many aircraft against many submarines while the best the Navy was able to do at the time was one-on-one. However the Navy did not want to use a GPSS model, their rationale being that it was not in FORTRAN. Running that model forced us to develop graphical outputs from data obtained during the running of the model. These graphs, plotted after the completion of the running, showed the positions of all the active elements as a function of time. Without this sequence of graphs it would have been impossible to convince potential users of the validity of the results.

Other groups in the Navy were more willing to use simulation as a tool to evaluate new ship designs and carrier operations. In 1965 Norden undertook two large-scale models, one to evaluate alternative systems for a new ship design that was eventually used to select a particular ship design and the components for that ship. The other model, started in 1965, has continued, as it evolved over the years, to be used to the present day. This model predicts the performance of the full complement of aircraft on board a carrier as they fly their different missions and require resources for repair, resupply, and spares.

These efforts for the Navy brought into sharp focus the weaknesses of the computer systems of the day and the implementations of GPSS. In order to run models with large data banks, it was necessary to gain control of the entire memory of the mainframe as subsets of data could not be stored on disk and brought into core memory as needed. This weakness continued even after the introduction of the System/360, as the early software still relied on magnetic tapes with some uses for disks. The advantage of the System/360 was that there was a chance, late at night, to utilize the entire machine for one simulation. Since GPSS was still at the time a bundled IBM product, we tried to interest IBM in making modifications to GPSS to enable large systems to be modeled on 360 machines with limited core memories. This need was the focus of getting users of GPSS to present their suggestions of improvements needed in the language to IBM. The approach was to use the established

organizational mechanism SHARE.

In the days of bundled software, IBM helped their users to cooperate through an organization, SHARE, to improve the use of IBM software and so increase hardware sales and to preserve their customer base. IBM users were encouraged to exchange software through the SHARE organization. SHARE held meetings twice a year, with very large attendance. SHARE was structured with numerous subgroups to focus on specific areas of user interest. Among the many subgroups was one addressing the use of discrete and continuous simulation. With the focus on bundled software, the SHARE simulation activities were concentrated on GPSS rather than SIMSCRIPT or SIMULA. There was considerable activity in the discrete event simulation area, as I can remember attending subgroup meetings in Chicago, Los Angeles, Houston, Miami, Toronto, and San Francisco.

At these SHARE meetings specific needs were communicated to IBM's marketing and programming staffs. The recommendations agreed to at these conferences resulted in the improvements of GPSS II, III, and 360. These efforts were funded by IBM in keeping with the then-current practice of maintaining IBM's lead over their competitors and in this manner contribute to maintaining their hardware sales. By the late 1960s there were GPSS simulator languages available or in the works from UNIVAC, Control Data, RCA, General Electric, Honeywell, and Scientific Data Systems, later to become Xerox. However, IBM's problems with software, in particular the operating system for the 360 series, limited the implementation of SHARE recommendations. Once IBM unbundled software products, further improvements in GPSS were not expected to be financially rewarding, so there was little incentive for IBM to develop significant improvements. It is an interesting aside that twenty-five years later, a desktop system can run the largest simulation and faster than those huge mainframes.

SHARE provided a limited opportunity to exchange information. It did not serve to establish a simulation literature. Tutorial books on how to simulate were still far off in the future. Papers offering advice to novices in the art of simulation had difficulty in finding a friendly editor. The editors were responding to the emphasis on the power of theoretical analysis to solve complex problems. Those working on complex problems that did not lend themselves to theoretical solutions still lacked organizational support. Simulation was still not an accepted method of analysis for these complex problems. Of course, those of us working in the field were obtaining results and very often quickly and cheaply using simulation. It was obvious to us that a symposium on discrete event simulation applications would find an audience.

A basic drawback to the prime role of SHARE in coordinating a conference on discrete event simulation was the exclusion of other hardware suppliers. Several of us addressed this problem and came up with the approach of establishing a larger coordinating organization to include in addition to SHARE, the Systems, Man, and Cybernetics and Computer Groups in IEEE, the simulation group in ACM, and any other organization that might have a kindred interest.

The choice of the Systems, Man, and Cybernetics Group—it was not yet a Society—was my doing. While at engineering school, I had joined the Institute of Radio Engineers (IRE) as a student member. After joining the commercial scene, young engineers were expected to join their professional societies. To further that goal, companies at that time encouraged membership by paying dues, travel expenses to go to evening meetings, and money for dinner. These meetings were interesting; and as I moved to new locations, there were new local subsections to form. In that manner, I met a number of people with similar interests in IRE. The experience I had obtained at Teleregister did not fit into the established IRE structure. When the American Institute of Electrical Engineers (AIEE) merged with IRE, those of us who had made note of this lack were invited to a meeting. Out of that meeting emerged the IEEE Systems, Man, and Cybernetics Group (SMC) in 1965. I was active in the formation of that organization, was on the ADCOM in 1967, and was its Chair in 1969. Under those circumstances, it is easy to see that when the concept of a simulation conference came up, as I already knew Dick Emberson at IEEE headquarters, the tactical approach was clear. Dick carried the ball for us by getting Bill Layton to lead us through the IEEE bureaucracy, which was not easy then; and I suspect is still not easy. Emberson pointed out the items that were needed to obtain Institute sponsorship after SMC committed their funds. George Moshos was active in getting the Computer Group to support the conference, though there was the feeling that they did not want to. There were some years that they failed to sponsor the conference.

Support from the Association for Computing Machinery (ACM) required some give-and-take, with Ralph Layer at ACM headquarters providing direction. Eventually, the Joint Users Group (JUG) acted as the mechanism to support the conference. ACM cosponsorship required a considerable amount of effort, some provided by John Lubin, as the meeting dates conflicted with the Fall Joint Computer Con-

ference.

IBM provided the critical help by allowing Arnold Ockene to spend considerable time establishing a structure for the conference. One measure of Ockene's success was that the registration fee was $30 in advance and $35 at the door for the two-day conference, luncheons included, at the New York Hilton. The Second Conference provided a *Digest* of papers, and the Third Conference achieved a true *Proceedings.* Some organizations that have later become sponsors refused to approve the initial conference.

Harold Hixson of the Air Force was coordinating the SHARE efforts in simulation, so it was a natural progression for him to be the first General Chair of what was to become the Winter Simulation Conference. I had the task of organizing the first program based on contacts from SHARE, IBM, IEEE, and ACM. The individuals who became active in the first conference came from industry, not academe. Partially this was the result of knowing where some things were being done. Another reason was to avoid the academic discussion of "my simulation language is better than yours," as new languages were being offered, at the user's risk, in great profusion. There had been conferences at different universities on this subject, and they had failed to produce a consensus. At this time the statisticians had not yet discovered the potential for analysis that simulation offered, so it was ignored. There was one major area of academic interest, determining the quality of pseudorandom number generators, and many papers on this subject were already in the literature and emerged in later conferences.

By 1967, Norden had gained considerable experience with GPSS and was able to present three papers on very different aspects of using simulation. Also, Norden with in-house funding and Navy support had modified the language to enable data banks to be stored on disk and brought into memory as needed. This activity had brought us into contact with other users of GPSS, and it was fairly easy to get them to contribute to the program. A total of thirty-five presentations were distributed over the two full days. Fourteen filled the plenary session on the first day, and the twenty on the second were in parallel morning and afternoon sessions with a panel discussion as part of the luncheon.

That panel discussion provided me with a keen insight into Geoffrey Gordon's approach and concept. He felt that the language should be so simple that the engineer could learn the language and use it without significant additional support aside from the manual. By then, I had enough experience modeling complex systems to be sure that was an impractical approach. GPSS was not used to model simple systems, like barber shops; instead, it was applied to extremely complex systems that did not lend themselves to available alternative theoretical solutions. I felt then and continue to feel that complex systems require skilled individuals, more likely collaborating teams, to produce a useful output. This has meant that simple systems have been made into simple models. Unfortunately, there has been the desire to model complex systems simply. It is, therefore, no surprise that in the end the results are both simple and useless.

The first conference program attacked complex system problems, some of which remain to this day. For example, the areas where delay occurred in the District of Columbia court system for criminal defendants were simulated and procedural changes suggested. This provided an example of a complex system that has been immune to ideas learned from simulation, as was also the case for operating procedures for emergency ambulances. On a more successful level, the simulation of handling of packages in an intermodal terminal of rail and trucks led to an improved new terminal.

One of the basic problems of simulation at that time, and independent of language, was the inability to observe during the simulation how the model was working. When a very large model was being run with many transactions and for a long period, it was particularly frustrating to find out after the "completion" of the run that there was an error at the beginning and all that considerable computer time was used to go through the same loop. We had that problem at Norden with the very large model being run for the Navy. This was an extreme case, as the input data, punched cards, were supplied by the Navy and loaded for each run. The question was how to determine if the punched-card deck did in fact introduce input data that the model could process, or if it was going to consume the allocated time, an hour after midnight, and then indicate an input-data error. This problem was made even more difficult since the Navy, for security reasons, did not allow Norden personnel to look through the readout and ascertain that the results were in fact "reasonable."

The solution to this problem was presented by Boeing (Huntsville), Norden, and IBM (Poughkeepsie). Those at Boeing (Huntsville) were the first to tie an interactive display terminal into a GPSS model and observe the results by interrupting the model on human command from the 2250 display unit. I had been invited to see their progress in 1965. The key was the 2250 unit, the first graphical device that could provide access to a simulation while the simulation was in progress. We at Norden used our 2250 for the same

purpose, interrupting the simulation, looking at critical block counts, and then allowing the simulation to continue till the next interruption or completion. While the 2250 was an expensive device, it did allow complex models to be observed while they were being run. This enabled much larger models to be built and debugged in a relatively short time. The world of the 2250 for the first time indicated what would happen when the model builder could debug the model online. The foundation for our current interactive desktop systems was laid at our very first conference.

The selection of New York as the site for the conference was influenced by several factors. Ockene was based in White Plains and could easily get to New York. Norden was in Connecticut, also close to New York, though the Program Chair had much less need to meet in New York. Details were worked out over the telephone with Harold Hixson. SHARE was accustomed to using hotels for conferences as they were too big for university facilities. There were a number of organizations in New York that would actively support the conference with personnel and by providing attendees. The New York Hilton was a new hotel and was looking to establish their conference business. So, we could arrange to hold the conference at that hotel with little advance planning. For this conference the costs did not seem to be out of line for a two-day meeting. The second, fourth, and fifth conferences also found New York to be affordable. An overnight single room for the second conference at the Hotel Roosevelt was $18.00. The big advantage New York provided at the time was the large attendance. The first conference ended up at full capacity, and some local IBM people were asked not to attend. Those funds collected from registration and not used for the conference were delivered to the sponsoring organizations as a reward. So began the tradition of each conference at least breaking even, although the usual case was to do much better. Part of the reason for this were the rules under which the sponsoring organizations agreed to advance monies for the conference. They always insisted that our attendance forecasts were overly optimistic, and almost always they were greatly in error. The budget for the first conference was based on a breakeven attendance of 200; there were actually 401 attendees.

After the first conference, I was promoted to be the General Chair of the second conference and Arnold Ockene became the Program Chair, the beginning of a tradition. We had not been happy at the Hilton, as the rooms for the plenary sessions had ceilings that were too low to allow the audience to read the slides being presented. Therefore, the second conference moved to the Roosevelt, an old hotel with high ceilings. It was a wise choice as the attendance was 856. Part of the reason for the much larger attendance was papers on any simulation language or any aspect of that language. The format that was to become characteristic of the Winter Simulation Conferences began to emerge.

Looking back to the early days of the conference once again reminds me that we were pioneers. We had faith that in spite of the prevailing attitudes, simulation would become a widely used tool—a tool that would enable system designers to understand more accurately the interactions in a complex system. Our early attempts to interact with the simulation while it is running are now routine. The color graphical outputs of today are beyond our very wild dreams. There are still areas that we expected to see more progress than has occurred. One such is real-time simulation following the real-time world to suggest quick emergency alternatives to the normal way of things, particularly in alternative controls for automobile traffic. So there are still areas for pioneering.

Finally, on a personal note, it has been a rare opportunity to contribute to an ongoing effort for so many years and see it grow and mature. Along the way there have been many people who have become friends and provided these fond memories. There have been very few unpleasant incidents, and for that we should be very grateful. Where simulation leads to in the future, I will leave for others.

JOSEPH M. SUSSMAN

See the article on the Twenty-Fifth Anniversary Keynote Address. In this panel discussion, I will elaborate on the themes that I discuss in the Keynote Address.

AUTHOR BIOGRAPHIES

MICHEL ARATEN is Vice President of Special Finance in the Real Estate Sector of the Chase Manhattan Bank. At Chase his responsibilities have ranged from workouts of real estate problem loans, corporate credit policy, setting up and running an insurance products group, and establishing a limited partnership mezzanine fund, to being director of the management science group. He was manager of operations research at Celanese Chemical and held operations research positions at Lever Brothers. He has held adjunct positions at Columbia University and Fordham University. He was Program Chair and General Chair, respectively, of the Fourth and

Fifth Conferences on Applications of Simulation. He holds the following degrees from Columbia University: B.A., B.S. in chemical engineering, M.S. in industrial engineering, and Ph.D. in operations research.

HAROLD G. HIXSON was born December 12, 1935. After education through local schools he graduated from Otterbein College (Westerville, Ohio) in 1957 with a B.S. degree in mathematics. Later he completed some graduate operations research course work at the University of Michigan. He received an Air Force commission as a Second Lieutenant through the Reserve Officers Training Corps. He spent the following $2\frac{1}{2}$ years on active duty and another $25\frac{1}{2}$ years in the active Air Force Reserve, retiring as a Lieutenant Colonel in 1985.

His first position was that of an Actuary at Wright-Patterson Air Force Base using the UNIVAC I computer. At the conclusion of his active duty commitment, he stayed on as a Civil Service employee in the same position. The actuarial work consisted of measuring the service life and forecasting logistics requirements for the Air Force's large inventory of aircraft engines.

His next position, starting in 1963, was that of a Mathematician, also at Wright-Patterson AFB, working as the technical monitor for several data-automation research contracts. This was also where he completed a study for the United States Civil Service Commission on the predicted effects of computer introduction on personnel and the consequent shifts in skills required over the next twenty years. In 1965 he started using GPSS (and later other simulation languages) as tools of operations research. A year later he was selected for a special course of full-time study in operations research at the Air Force Institute of Technology. After completing this, he became an Operations Research Analyst in the Air Force Logistics Command's new Simulation Center. One of his first assignments was to teach the simulation portion of the "System Analysis and Operations Research Seminar" for the Dallas Region of the Civil Service Commission. He succeeded Phil Kiviat as the System Simulation Project Manager for SHARE in March 1967. He was General Chair of the first Winter Simulation Conference (then called the Conference on Applications of Simulation Using GPSS) in November 1967. He was also General Chair of the first Summer Computer Simulation Conference (then called the Conference on Applications of Continuous Systems Simulation) in June 1969.

As Project Manager for SHARE he conducted a GPSS User Survey of desired features, which constituted a major determinant of the feature content of the GPSS V language. With Norm Nielsen, he taught computer-system simulation courses for Simulation Associates, Incorporated at major cities in the U.S. and Canada during the early 1970s. With William Drake, he ported the Extendible Computer System Simulator I to Honeywell mainframes in 1974.

He has continued to work as an Operations Research Analyst (since 1975) in the Management Sciences Directorate of the Deputate for Plans and Programs in the Air Force Materiel Command. His most recent work has been to embed simulation models in decision support systems.

AUSTIN C. HOGGATT is a Professor Emeritus of Business Administration in the University of California, Berkeley. In 1956, he constructed in machine language on the IBM 701 at Poughkeepsie, the first computer simulation of an economic model (Hoggatt 1959). With Fred Balderston and nine man-years of labor, he built a FORTRAN simulation of the market process on the 709 at the Western Data Processing Center, UCLA (Hoggatt and Balderston 1962). Thereafter, for years in the wilderness he preached the virtues of APL with few converts. He recently returned to FORTRAN with a simulation study of teams of RISC workstations solving a large chemical-lattice problem (Hoggatt, Dawson, and Visser 1990). While this work showed one-tenth the computation time and one-tenth the capital cost versus the IBM vector-capable supercomputer at Berkeley, and could have been implemented using a patent by Martin Graham (Department of Electrical Engineering and Computer Science, University of California, Berkeley), we were not successful in turning the establishment away from fixation on their expensive monolithic machines.

In the 1960s he worked with a group of management scientists to develop the Berkeley Laboratory for Man-Machine Simulation (Hoggatt, Escherick, and Wheeler 1969). The first published experiment in oligopoly theory (Hoggatt 1959) was followed by a series of experiments with human subjects embedded in models from economic theory. After spending large sums of NSF money to capitalize this facility, the University of California, Berkeley scrapped it for want of funds to pay for maintenance.

Now, the knowledge and expertise developed in the design and use of the CRMS behavioral laboratory is directed into field studies of energy use in apartments in Eastern Europe (Hoggatt et al. 1991). This is a significant social problem in that the housing stock of Eastern Europe is grossly energy inefficient and makes a large contribution to the worldwide carbon

burden. The intent is to utilize inexpensive computers on a chip with inexpensive transducers to monitor energy use and associated human behavior with the intent of inducing energy-conserving behavior to complement engineering energy retrofits to these buildings (Hoggatt and Wasik 1992).

PHILIP J. KIVIAT is Vice President, Federal Programs for KnowledgeWare, Incorporated, the world leader in Computer Aided Software Engineering (CASE) application development and reengineering tools, where he is responsible for KnowledgeWare federal sales and support operations. Before joining KnowledgeWare, he was Senior Vice President of ICF Information Technology, Incorporated, an operating unit of ICF International, Incorporated, where he was responsible for ICF Infotech's Information Engineering, Technology Management, and Decision Support Systems business units and operated Infotech's federal Information Resources Management consulting practice. Prior to his joining ICF, Mr. Kiviat was a Vice President at Chartway Technologies, where he managed a consulting practice specializing in federal acquisition, management, and organization of information-technology-driven organizations, and software-technology R&D. He was previously responsible for Chartway's marketing and sales activities, and was Division General Manager when Chartway was a division of SAGE Systems, Incorporated. He has thirty years of technical and managerial experience in information-systems design and implementation, in the management of high technology organizations, and in management systems engineering.

Before joining Chartway, Mr. Kiviat was corporate Vice President of CTEC, Incorporated. He was previously Client Manager of the Washington Office of SEI Information Technology and Technical Director of the Federal Computer Performance Evaluation and Simulation Center (FEDSIM). He has also been associated with Systems Control, Incorporated (Division Manager), Simulation Associates, Incorporated (President), the RAND Corporation (Member of the Technical Staff), and the United States Steel Corporation. He is the author of two books on simulation programming languages (SIMSCRIPT II and GASP) and numerous technical papers. Mr. Kiviat has given over two hundred invited speeches and presentations to technical and management groups worldwide.

In 1974 Mr. Kiviat received a special ACM professional award in recognition of his contributions to the use of computers within the federal government. In 1976 he received the Senior Executive Award of Excellence from the Interagency Committee on Automatic Data Processing, and the prestigious A. A. Michelson Award from the Computer Measurement Group for his contributions to computer performance management. During 1978, Mr. Kiviat was a Task Group Leader for President Carter's Data Processing Reorganization Project. He was later a member of the Advisory Committee of the Grace Commission. In 1988 Mr. Kiviat was inducted into the Government Computer News Information Resources Management Hall of Fame; and he was elected in 1990, 1991, and 1992 to receive Federal Computer Week's Federal 100 Executives award. He has been a member of the OMB Information Technology Advisory Committee, a consultant to the GAO, and a member of the National Research Council Panel for the assessment of the programs of the Computer Systems Laboratory of the National Institute of Standards and Technology. He has been the Program Chair of the Industry Advisory Council of the Federal Government Information Processing Councils and is presently FGIPC Delegate for the Council.

Mr. Kiviat has degrees in mechanical engineering and industrial engineering from Cornell University, has completed his course work for a doctorate in business administration at the University of Southern California, and has attended the Federal Executive Institute.

MICHAEL F. MORRIS is the Director of Productivity Management Center and senior management consultant. He has more than thirty years of experience from entry through executive levels in computer-and-communications-systems research, planning, acquisition, development, implementation, and use in government and private sector installations and in scientific and business computing environments. His main expertise areas are: capacity planning; performance analysis and improvement; mathematical modeling of complex systems; program management (recruiting, training, organizing, motivating, and supervising technical staff); and developing and documenting high-technology systems and projects. He has analyzed more than one hundred computer installations; developed office-automation systems and facilities; performed quality control and assurance functions for systems, services, and production; and trained and developed capacity planning and performance evaluation teams. He helped develop and managed the world's largest real-time computer network for suborbital tracking and impact prediction during missile launches and for calculating ephemerides of artificial earth satellites. He automated calculation and drawing of real-time displays used to determine when to destroy errant missiles. He designed and oversaw implementation of the Apollo Launch Abort

System to track and rescue astronauts during the early launch phase. He supervised technical groups of up to 450 and commanded up to 700 Air Force personnel.

Mr. Morris wrote the book *Computer Performance Evaluation: Tools and Techniques for Effective Analysis* (Van Nostrand Reinhold, 1982), and he has contributed many articles to trade and professional publications. He served as editor and publisher of *Simuletter*, *Computopics*, and *Performance Evaluation Review*. He was consultant to the publisher and the first contributing editor during the development of *Government Computer News*. He planned, wrote, produced, and hosted a television interview series for *Government Computer Expo*.

Mr. Morris earned the M.B.A. degree in production management and the B.S. degree in mathematics at Michigan State University. He graduated from the Russian Linguist Program of the AF Institute of Technology at Syracuse University, and he completed the International Relations Program at Cambridge University in England. He also studied aeronautical engineering at Wayne State University in Detroit and orbital mechanics at the University of South Florida.

Mr. Morris is a member of the Armed Forces Communications and Electronics Association and the American Institute for Aeronautics and Astronautics. He was the Chair of the Federal Interagency Computer Performance Evaluation Users Group and Chair of the Interagency Task Group on Computer System Performance for the National Bureau of Standards. He was Director of the Association for Computing Machinery's Special Interest Group on Performance Evaluation, Chair of the Special Interest Group on Simulation, and the first Fellow of the Institute for Software Engineering. He has lectured for American Management Association, Association for System Management, Data Processing Management Association, Institute of Internal Auditors, U.S. Professional Development Institute, and others. He was appointed for three terms as a Faculty Advisor on Data Processing Curricula for the State University of New York. He organized, chaired, publicized, or served as program chair for a variety of professional and commercial workshops, seminars, conferences, and expositions attended by up to fifteen thousand people.

ARNOLD OCKENE was one of the organizers of the first Conference on the Applications of Simulation Using GPSS in 1967; and in the following three years, he was, respectively, the Program Chair, the General Chair, and Arrangements Chair of the Second, Third, and Fourth Conference on Applications of Simulation. (The Fifth Conference on Applications of Simulation was also known as the 1971 Winter Simulation Conference.) From 1961 to 1969, Mr. Ockene was involved in the evolution of GPSS in IBM, first as a user and teacher, then as IBM's interface between customer users and the program developers. After a five-year hiatus he returned to IBM where he spent ten years with IBM World Trade, six in its Academic Information Systems unit, and the most recent two years in technical computing.

JULIAN REITMAN teaches the history of science and technology at the University of Connecticut, Stamford campus. Previously he taught at George Mason University (1988–1990), the University of Bridgeport (1972–1976), and New York University (1970). He received a B.E.E. degree from the City College of New York in 1949, and he received an M.E.E. degree from New York University in 1954.

From 1955 to 1961, Mr. Reitman worked at Teleregister Corporation, performing systems analysis and design for airline-reservation systems. He also developed simulation models of airline-reservation traffic in assembly language on an IBM 650 computer.

During the period 1961–1987, Mr. Reitman worked at Norden Systems, performing system analyses of complex systems using discrete event simulation languages, primarily GPSS. He also developed, used, and made available an enhanced version of GPSS called GPSS/Norden with a man-machine interface, improved graphics, and data bases. The areas of application of GPSS/Norden included:

- Computer-and-communications systems which combine message switching and data processing;
- Transportation systems for Urban Personal Rapid Transit (PRT), intercity surface mass transit, and FAA air traffic control;
- Systems effectiveness of airborne and shipboard systems;
- Performance prediction for surface-to-air missile systems;
- Analysis of built-in tests for radar systems;
- Traffic capacity of centers processing income tax returns;
- Data processing and yield-prediction system for integrated circuit production; and
- Performance prediction for word-processing systems.

Mr. Reitman has conducted and aided in establishing and implementing computer systems for a variety of applications with U.S. organizations as well as organizations in Australia, Canada, Germany, Italy, Israel, Japan, the Netherlands, the People's Republic of China, and the United Kingdom.

Mr. Reitman is the author of *Computer Simulation Applications* (Wiley, 1971), and he has written a chapter on simulation in *The Handbook of Computers and Computing* (Van Nostrand, 1984). He has also written numerous articles for *Simulation*, *IEEE Spectrum*, and *Computer* on system-simulation applications, simulation-language enhancement, evaluation of simulation results, computer-system design, man-machine interface for simulation, and integration of computer graphics with simulation.

Mr. Reitman is a Senior Life Member of IEEE. He is also a member of the American Association for the Advancement of Science and the Society for the History of Technology.

Mr. Reitman helped start the Winter Simulation Conference, serving as the Program Chair of the Conference on Applications of Simulation Using GPSS in 1967 and then as the General Chair of the Second Conference on Applications of Simulation in 1968. He served on the Board of Directors of the Winter Simulation Conference from 1971 to 1985, representing the IEEE Systems, Man, and Cybernetics Society.

JOSEPH M. SUSSMAN is the JR East Professor and Professor of Civil and Environmental Engineering at the Massachusetts Institute of Technology. He has been active in the field of systems analysis and simulation applications and methodologies as applied to transportation for many years, focusing in the areas of rail, freight, and intelligent vehicle highway systems. He has worked with a number of transportation organizations, government agencies, and commissions in the U.S. and abroad; and as the first Distinguished University Scholar at IVHS America, he has helped structure a national program in this new area.

Professor Sussman was the Program Chair of the 1971 Winter Simulation Conference, and he was the General Chair of the 1973 Winter Simulation Conference. He has authored many publications, has lectured extensively in the U.S. and abroad, and has served as the Head of the Department of Civil Engineering and as the Director of the Center for Transportation Studies at MIT.

JAMES R. WILSON is a Professor in the Department of Industrial Engineering at North Carolina State University. He received a B.A. degree in mathematics from Rice University in 1970, and he received M.S. and Ph.D. degrees in industrial engineering from Purdue University in 1977 and 1979 respectively. Previously he worked for the Houston Lighting & Power Company (1970–1972), the U.S. Army (1972–1975), the University of Texas at Austin (1979–1984), and Purdue University (1985–1991). His current research interests are focused on the design and analysis of simulation experiments. He also has an active interest in applications of operations research techniques to all areas of industrial engineering. He currently serves as an Associate Editor of *IIE Transactions*, Departmental Editor of *Management Science* for Simulation, and Program Chair of WSC '92. He was Associate Program Chair for WSC '91 and *Proceedings* Editor for WSC '86. He has also held various offices in TIMS/College on Simulation.