Proceedings of the 2004 Winter Simulation Conference
R .G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, eds.

A VIEW FROM THE BEGINNING: WHEN DOES A DESCRIPTION BECOME A TAXONOMY

Philip J. Kiviat

11202 Tildencrest Court
Guerra, Kiviat, Flyzik & Associates, Inc.
Potomac, MD 20854-2746, U.S.A.

ABSTRACT

This paper describes the author's career leading up to the publication of his 1969 paper *Digital Computer Simulation: Computer Programming Languages*, how it influenced the paper, and why the paper has endured as a taxonomy for discrete-event simulation programming languages.

1 INTRODUCTION

Looking back some 46 years to when I first got involved with simulation it is hard to remember major milestones, much less details about why I did this or did that or why such and such happened. Nevertheless, I will try to recount how I came to be involved with discrete-event simulation and the events that led up to my writing The RAND Corporation Memorandum RM-5883-PR, *Digital Computer Simulation: Computer Programming Languages* that was issued in January, 1969.

2 THE BEGINNING

Not being able to get a "real job" the summer of 1958 I took a position as an assistant to Professor Robert Bechhofer at Cornell University doing Monte-Carlo sampling simulations to support his experimental statistical research. I learned about random numbers and pseudo-random numbers and how to generate them, and started thinking about what it meant to model something and to perform artificial experiments. From this modest work, being the only student at the university at that time doing anything like this, I became known as "the simulation expert." Hardly true, but a nice credential nevertheless.

From this summer work I progressed to a part time job in the Cornell Computer Center supporting research projects, doing Monte-Carlo modeling for such projects as population genetics. As a student I was learning about job shop scheduling simulation from Professor Richard Conway and then graduate assistant Bill Maxwell.

I added ALGOL to my repertoire of programming languages (I knew FORTRAN, COBOL and various assemblers) and started thinking seriously about how language concepts and structure influence how people think about solving problems.

After getting my Masters Degree in Industrial Engineering, having some proficiency in FORTRAN, ALGOL, COBOL and assembler languages, and an overrated reputation as a "simulation expert" I was ready to enter the job market.

In June, 1960 I went to work at the United States Steel Applied Research Laboratory in Monroeville, PA, having been hired to lead a team of Industrial Engineers to build simulation models of open-hearth steelmaking plants that would allow the engineers to study how best to schedule facility and equipment operations to maximize plant steelmaking output.

It was here that my exposure to industrial simulation needs began and my thoughts turned to "industrial strength" as opposed to "academic" modeling issues.

3 FIRST PRINCIPLES EMERGE

Before I could build a model, or indeed, even think about one, I needed to understand how an open-hearth steelmaking facility worked. To do this I visited several US Steel plants, had the plant operations described to me by plant industrial engineers, and toured the facilities to see them in action. I watched as steel was "cooked" and then poured, as cranes moved up and down overhead tracks to service them, and as various "overhead" functions were performed to keep equipment in service and provide raw materials. Then, to develop a logical discrete-event model that could be simulated to reproduce the statistically varying production of a plant, I investigated what others in similar businesses were doing at the time. In 1961 there wasn't much to look at and learn from, but conversations with colleagues about the modeling and programming language work that was going on in Great Britain (K D Tocher's GSP work at the United Steel Companies) and in the US

(Geoffrey Gordon's GPSS and Harry Markowitz's SIMSCRIPT) were very informative and set me thinking in the right track.

I decided that my first need was to create a modeling language that allowed me to communicate with the industrial engineers I was working with, none of whom were computer programmers and few of which were analytical operations research analysts. Without a discussable dynamic model of plant operations I couldn't go forward. I couldn't talk about representations of physical objects, I couldn't talk about system dynamics, I couldn't explain how modeling rules and logic governed actions that took place in a dynamic system, and I couldn't explain how statistical variability was introduced and controlled. The modeling language had to be understandable by engineers so we could build models, discuss them, and create scenarios for their use.

After thinking about the relative merits of activity scan, event scheduling and transaction flow modeling I decided on event scheduling. This was partly because it was easy to understand and completely transparent, and partly because I needed a modeling language I could put into use immediately. (While US Steel Research was farsighted enough to fund my modeling research and programming language development, its interest was in having simulation models that could be used to improve plant operations, not in sponsoring simulation research.)

I decided to build my own simulation programming system using FORTRAN as a base language with the discrete-event modeling concepts and constructs implemented through functions and subroutines. I built it as I worked through the problems of modeling open-hearth steelmaking plants, learning as I went about what was needed for modeling and what was necessary for model implementation. Thus was developed GASP, the General Activity Simulation Program.

Note: An activity in GASP is bounded by two events that are the starting and ending points of the activity in simulated time. GASP deals with events explicitly and activities implicitly. GSP, the activity based language created by KD Tocher, deals with activities explicitly through an activity scanning mechanism. In GSP the activity and not the event is where the logic of a model is represented.

GASP worked, and the industrial engineering team and I successfully built, demonstrated, and experimented with simulation models of several active US Steel open-hearth steelmaking plants.

Looking back over that experience I learned that the most important feature of a simulation language is that it express and communicate modeling-oriented thoughts and concepts. A suitable language had to be able to define and manipulate permanent and temporary objects, objects that were passive and objects that exhibited active behavior; it had to be able to express system dynamics and the rules that govern system behavior; and it had to be able to reproduce the variability in the nature of objects and how they interact that is seen in the real world.

I also came to the conclusion that while you could implement a modeling language in a general purpose programming language such as FORTRAN, the clarity of the model was often so obscured by the programming baggage of the implementation that a specialized simulation programming language would do a far better job of both shaping a modeler's thoughts and communicating them.

4 EVOLUTION

In 1963 I moved to The RAND Corporation to work with Harry Markowitz on the second generation of SIMSCRIPT, SIMSCRIPT II. Being an event-oriented language, SIMSCRIPT was conceptually similar to GASP. The SIMSCRIPT II project added modeling power to SIMSCRIPT and used its implementation as a modern programming language to build in simulation features and capabilities that went far beyond the capabilities of a FORTRAN package or FORTTRAN generator.

While at RAND I participated in many conferences that brought together people doing simulation language research in the US and Western Europe. From 1963 through the early 1970's people who had "made their mark" in the programming language community and in the simulation language community gathered together in various venues to share what they were doing and advance new points of view. The people who took part in what became over the years a periodic "gathering of luminaries" included: KD Tocher, John Laski, John Buxton, Bob Parslow, Robin Hills, Edjer Dykstra, Tony Hoare, Christopher Strachey, Nicholas Wirth, Ole Johan Dalhl, Kristen Nygaard, Alan Clementson, Pat Blunden, Howard Krasnow, Geoff Gordon, John McNeley, Donald Knuth, Tom Naylor, Alan Pritsker, Julian Reitman, Tom Schriber, David Parnas and others. Simulation modeling, programming and statistical analysis ideas were proposed and dissected, ideological camps were formed (event, activity, process-orientation) and many advanced programming language ideas were discussed in the context of their use for simulation purposes. It was a time of great intellectual excitement for me with a mingling of minds from very different backgrounds and perspectives, most academic but some commercial and industrially-focused.

By the time this period ended and the simulation community focused what is now the Winter Simulation Conference as the gathering place for simulation practitioners and researchers, things had pretty well settled down into well understood patterns:

- The simulation community had pretty much settled on Activity-oriented languages, Event-oriented languages, and Process- oriented languages as paradigms for modeling system dynamics. Harry Markowitz's term "World View" expressed well

how a modeling paradigm captures the essential elements of how the modeler's world works.

- Programming languages were being designed/improved/implemented to host one or more of these modeling approaches. Great strides were made in programming language constructs and the best minds of the time took part in conferences devoted to new programming language ideas.
- Languages were being extended to deal with data structures for static system modeling and representation (e.g. entities, attributes and sets), model management, statistical analysis and experimentation requirements.

By the fall of 1968 I saw relative stability in the simulation concept world and thought it would be worthwhile to sort out my thoughts on the modeling and programming of discrete-event simulation models. I did this as I was still embarked on the development of the SIMSCRIPT II programming language and thought that going through the research/writing process might open up some new avenues that might improve the language, and to share my thinking with others in the simulation community.

Digital Computer Simulation: Computer Programming Languages was published as one person's description of what he saw happening in the discrete-event simulation community, and not as a suggested, much less proposed, taxonomy. When I was invited to prepare this paper I was told that "[my] 1969 paper still provides the fundamental taxonomy for simulation modeling - all modeling thinking builds from it." When I declined, it was offered that upon re-reading the paper, it appeared that I was not proposing a strict scientific taxonomy at all - it just became such. That's exactly what happened.

5 LOOKING BACK

The last two sections of my paper were titled "Current SPL Research" and "The Future of SPLS".

In the penultimate section I discussed research then going on in Simulation Concepts, Operating Systems, Interactive Languages, Time-Sharing, and Graphics. It was primitive, as the technology in use then was far less capable in almost every sense than the technology we have today. But it was on track. As one might expect, today's simulation languages and the web make possible things that were barely thought of then.

In my final section I said that the greatest challenge to the simulation community (in 1969) lay in the unification of discrete-event and continuous-event simulation languages.

When preparing this paper I looked at *A Collection of Modelling and Simulation Resources on the Internet* published by Andrea Emilio Rizzoli of IDSIA and saw that this challenge had been met and answered with three languages dealing with hybrid simulation and over 90 simulation languages and packages catalogued, some for languages and kinds of simulation that were not even thought of then.

The research and industrial world has been busy for the past 35 years adding to the store of simulation knowledge and experience and successfully adapting the ideas of my 1969 paper -- which has become taxonomy -- to the hardware and software technologies that come and go.

Looking back I guess we got it right.

Looking ahead I see the essential simulation modeling concepts I described being preserved -- active and passive entities will continue to be viewed as acting on one another through rules and logic described as events, activities or processes -- while the languages used to describe them continue to flourish and become more elegant, more powerful, more intuitive, more visual, and hopefully, more easily used.

REFERENCES

Kiviat, Philip, 1969, *Digital Computer Simulation: Computer Programming Languages*, The RAND Corporation, Memorandum RM-5883-PR

AUTHOR BIOGRAPHY

PHILIP KIVIAT is a founding Partner of Guerra, Kiviat, Flyzik & Associates, Inc., a strategic marketing and sales consulting firm specializing in solution selling, sales strategy and tactics, and market analysis and positioning. He was the first Technical Director of the Federal Computer Performance Evaluation and Simulation Center (FEDSIM) that was established by the US Air Force in 1972. He is the author of two books on simulation programming languages (SIMSCRIPT II and GASP) and numerous technical papers. The "Kiviat Graph" is widely used in the computer hardware and software performance field, as well as in other disciplines, to portray performance patterns and distinguishes modes of behavior. In 1974 he 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 prestigious A. A. Michelson Award from the Computer Measurement Group for his contributions to computer performance management. In 1988 he was one of the first inductees to the Government Computer News Information Resources Management Hall of Fame, and was elected in 1990, 1991, 1992 and 1999 to receive Federal Computer Week's Federal 100 Executive award. He has degrees in Mechanical Engineering and Industrial Engineering from Cornell University and completed his course work for a Doctorate in Business Administration at the University of Southern California. His e-mail address is <pkiviat@govspecialists.com>.