Proceedings of the 1984
Winter Simulation Conference
S. Sheppard, U. Pooch, D. Pegden (eds.)

MODEL DEVELOPMENT REVISITED[*+]

Richard E. Nance

Systems Research Center
and
Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

ABSTRACT

A chronology of simulation language development provides the background for understanding the current status of simulation model development. Factors characterizing the current status include a shift in emphasis from program to model, more commitment to modeling tools, and the lingering impedance of simulation language isolation. Current and future needs are identified. Specific approaches to meeting these needs are cited in an extensive description of current research, and a brief review of current perceptions of software development technology portends a convergence toward the paradigm set forth by the model based methodology. We conclude that the technology of simulation model development continues in a transition that portends more rapid changes for the future.

A BRIEF HISTORY OF SIMULATION SOFTWARE

A brief chronology of simulation software conveniently divides into five periods: the early era of custom programs, the period of emergence of simulation programming languages (SPLs), the second generation of SPLs, the era of extended features, and the current period.

During 1955-60, simulation like most computing applications was done with custom programs, i.e. each simulation required the development of all software necessary for accomplishing that task. The late K.D. Tocher lay the groundwork for changing this with his recognition of common functions, grouped together under the title General Simulation Program (GSP) [43]. Tocher's contribution of GSP, and his writing of the first book [42] contributed much to the early realization of the importance of software support for the simulation task. Tocher also invented the Wheel Chart, a forerunner of the Entity Cycle Diagram, which provides a conceptual basis for symbolic modeling underlying the program generators still in use in the United Kingdom and elsewhere.

----

* This paper is an updated and expanded version of "A Tutorial View of Simulation Model Development, *Proceedings of the 1983 Winter Simulation Conference*, pp. 325-331.

+ Research contributing to this paper was supported in part by the U. S. Navy under Contract No. N609201-83-G-A165 through the Systems Research Center, Virginia Tech.

The first SPLs emerged during the 1960-65 time frame. Thorough histories of both GPSS [9] and SIMULA [33] are available. Control and Simulation Language (CSL), produced by Buxton and Laski [5] in the UK, and the first version of GASP was developed by Kiviat [15]. Interestingly, the software developed during this five year time period forms the foundation for the simulation software in use today.

The second generation of SPLs followed in the time frame 1966-70. GPSS II, III, 360, and V all appeared in this period as did several versions of SIMSCRIPT II - II.5, and II-Plus. SIMULA 67 superseded the earlier version, and Extended CSL (ECSL) replaced its ancestor. Simulators like GASP took on various new forms as well; e.g. GASP II, IIA, and others.

While entirely new issues of SPLs were uncommon in the 1971-78 period, marketing strategies emphasized the addition of features to the existing versions. For example SIMSCRIPT II.5 incorporated the process concept and added a continuous simulation capability. GPSS shed some of its insularity and enabled external access to FORTRAN and PL/I routines. In an ambitious effort at Norden, actually begun in the late 1960s, graphical abilities were added in a version permitting limited user interaction, designated as NGPSS [32]. The interactive versions of other SPLs began to appear toward the end of this period.

Major developments in the UK and Europe during the 1971-1978 period extended the ideas introduced with Programming By Questionnaire [35,36] to the interactive production of simulation programs. Prominent in this work are the original contributions of Clementson [6] in the development of CAPS based on ECSL, the multiple target language capabilities of DRAFT [23,24], and the modular design suggested with MISDES [7]. Related efforts, with more ambitious goals in the U.S., are described in the papers by Heidorn [11,12].

Toward the end of this period, concerns for more fundamental issues in simulation modeling appeared in the book by Zeigler [45], which drew together ideas published earlier in various papers and reports. At the same time, the need for a better domain for model development appeared in the work of Nance [27], Kleine [17], and Oren [37]. Efforts such as Nelson and Lindstrom [31] and Heimberger [13] began to illustrate the significant capabilities for interactive model development and program execution.

THE CURRENT STATUS

Simulation model development is in a transition period: the transition in focus from programming to model development. This transition is reflected in the interest and activities of organizations ranging from marketing firms such as Pritsker and Associates to research groups in universities. While several factors characterize the transition, three are most obvious:

(1) a shift from the *program* to the *model* view of the simulation process,

(2) interest in and commitment to the development of support tools, and

(3) the influence of a concept/language impedance.

The shift in focus from program to model is reflected in the increasing concern for conceptual problem description in contrast with language prescribed guidelines. The Graphical Modeling and Simulation System (GMSS) is one example [2], and recent extensions of program generators [25,26] offer yet another. In one sense the model view represents a realization that executable languages often are constraining in their realization and expression of concepts, and the "rush to code" is a poor design strategy. Stemming from this emphasis on conceptual modeling is the development of intermediate specification forms, most often not executable in themselves. The Ship Combat System Simulator (SCSS) [39] utilizes a network representation with combat system elements described as nodes following a specific syntactic format. The nodal definition and the linkages among nodes prescribed in SCSS provide a semantic structure closer to the conceptual views of the combat system engineer than can be derived from the SIMSCRIPT II.5 code, that constitutes the eventual (executable) representation. Other examples can be cited to support the claim that multiple model representations are becoming more the "standard" for large, complex models, and the clear trend is toward the separation of model description and program execution.

Increasing expectations indicated by the use of simulation for yet larger and more complex models and the increased focus on model description have ushered in new concerns for tools to support the model development process. Commercial products now offer auxiliary data base systems and graphical output generators. The communication and formatting capabilities of SDDL [18] are being augmented by analysis routines that are applied to non-executable model representations. The benefits to be derived from utilizing formalisms, based on modeling rather than programming needs, for simulation support are becoming more apparent [46,47]. Such support tools will play major roles in the *verification* of non-executable model representation.

The concept/language impedance stems from the parochialism created by slavish adherence to SPL representations of world views, see [29] for further discussion of this problem. Even more serious is the continued use of general purpose languages, in preference to SPLs, for simulation modeling. Despite the optimistic expectations of educators, no decrease is readily apparent in the number of models in FORTRAN, PL/I, PASCAL, etc. This fact, perhaps more than any other single point, emphasizes the perceived difficulties of translating modeling concepts into a correct SPL representation. Nevertheless, the barriers of language isolation will continue to inhibit the development of simulation model representation. As Kiviat [16] aptly phrased it so many years ago, we continue to have an "inversion of theory and interpretation" with the misguided view that the theory is expressed by an SPL.

THE MODEL LIFE CYCLE

Figure 1, taken from Nance and Balci [30], characterizes the model life cycle as progressing through chronological periods: problem definition, model development, and decision support. Figure 2 offers an elaboration of the phases within each of these periods and depicts the processes by which a modeling study transitions from one phase to another.

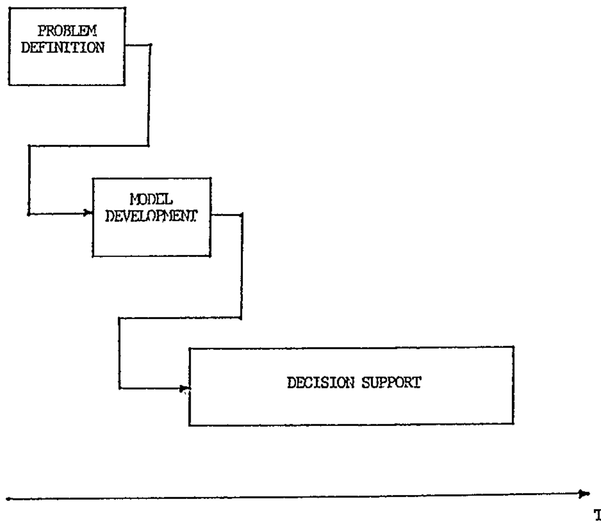

Figure 1. The Chronological Periods of the Model Life Cycle.

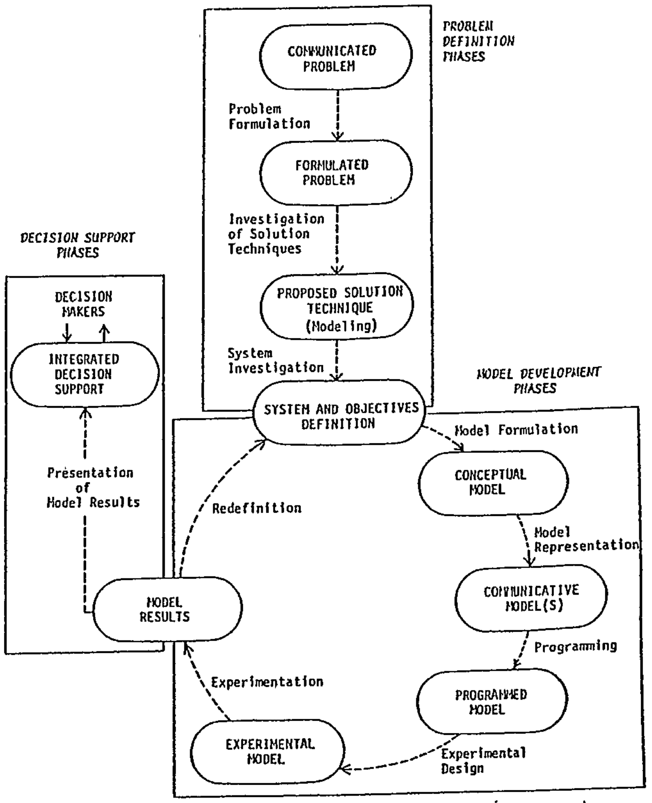

Figure 2. Phases in the Chronological Periods of the Model Life Cycle.

The activities during the problem definition phases principally involve the "client" and project manager dialogue that hopefully results in a precise definition of the system to be studied and the objectives to be realized from the study. Problem definition is dependent on both technical and organizational (political) factors, and success can be achieved only by effective communication among the participants and the documentation of decisions reached during these phases.

The model development phases begin with the defined system and the stated objectives. Conceptual models in the minds of one or more modelers must eventually find expression in one or more communicative models. The communicative model represents a basis for assertions and tests as well as the reconciliation of varying concepts. The program model follows from a communicative model; and, embodied within an experimental design, the experimental model produces results. Note that verification is intended to be used wherever possible in all of the phases of problem definition and model development. Validation in the traditional sense is reserved to the comparison of model results with system behavior after completion of the experimental model.

The integrated decision support period is initiated with the acceptability of the model by the client manager(s). Again, both technical and organizational factors can contribute to the acceptance decision; however, the support tools can contribute significantly to the model credibility, which is considered to be the most crucial factor in the acceptance decision.

FUTURE GOALS FOR SIMULATION MODEL DEVELOPMENT

The most comprehensive goal expressed by researchers in simulation modeling is the creation and consequent realization of the Model Development Environment (MDE). The MDE would provide an interactive setting for model creation so that the modeling activities, supported by necessary model development tools, contribute to long term organization assets in the form of models, data, experimental designs, and experimentation results. An analyst or modeler, within the MDE, would be supported in a structured, more axiomatic approach to the modeling and experimentation activities. Model verification, supported by such tools, would be applied throughout the model development phases. Emphasis in the early model development phases would be on problem definition and precise statements of system boundaries and study objectives. Only later would the issues of efficient execution emerge as constraints as decisions are reached regarding the implementation of executable model representations.

A second important goal is that support be provided throughout the model life cycle. Of course, this goal is intimately linked to the first.

APPROACHES TO THE IMPROVEMENT OF SIMULATION MODEL DEVELOPMENT

The intent of this section is only to identify approaches to improvement. References are provided so that the interested reader can consult them for details and specific information. The approaches are categorized as follows: (1) extension of software development techniques, (2) extension of program generators, (3) extension of SPL definition, (4) system specification languages, and (5) model-based methodology.

Some claim that simulation modeling is only a minor extension of programming in software development. Consequently, the Programming Support Environment (PSE) or Software Engineering Environment (SEE) provide all of the necessary tools. Perhaps a counter example to this opinion is found in the necessity for creating SDL/SDA as an extension of PSL/PSA [40,41] for simulation applications. The relationship between programming and model development is discussed further in the following sections.

The program generator technology is widely used in the United Kingdom and elsewhere in Europe. Some program generators such as DRAFT and CAPS are now rather mature software systems. Extensions to these generators are viewed as providing ready communication between management and analyst, and some capabilities for decision support are believed to be readily achievable if not already present in current versions [26].

One school of thought is that more formal modeling approaches are required to deal with the complex challenges of simulation applications [47]. General systems theory is viewed as providing the foundation for such approaches [38]. SPLs "based explicitly on systems theoretic concepts" and the "development of conceptual and mathematical theories for guiding the practice of modelling and for designing software tools ..." offer advantages over current approaches [38, p. 70]. Also within the scope of SPL extensions, but differing from the general systems theory approach is the Entity-Attribute-Set (EAS) structure suggested by Markowitz [21]. Utilizing the current five levels of SIMSCRIPT II.5, Markowitz extends the language applicability to a data base level and beyond. The result is a more powerful descriptive mechanism but one that is still executable. An application development system based on EAS and utilizing nonprocedural language database inquiry is a recent product of this approach [22].

The Delta Project [14], cooperatively between the Norwegian Computing Center and the University of Aarhus, represents a holistic view of life cycle support. While the Delta Project can be viewed in a narrow sense as another system specification language, the philosophy advanced by Nygaard and Handlykken [34], [34] reflect an intent much broader in scope.

A final approach is the model based methodology, which is descriptive of the Conical Methodology (CM) [28]. This methodology forms the basis for an implementation of a Model Development Environment that is illustrated in Figure 3, taken from [3]. The structure of the Ada* Programming Support Environment [1] is followed in explaining the support tools for modeling. The CM emphasizes the hierarchical decomposition through a top-down model definition followed by a bottom up model specification.

----

* Ada is a registered trademark of the U.S. Department of Defense Ada Joint Program Office.

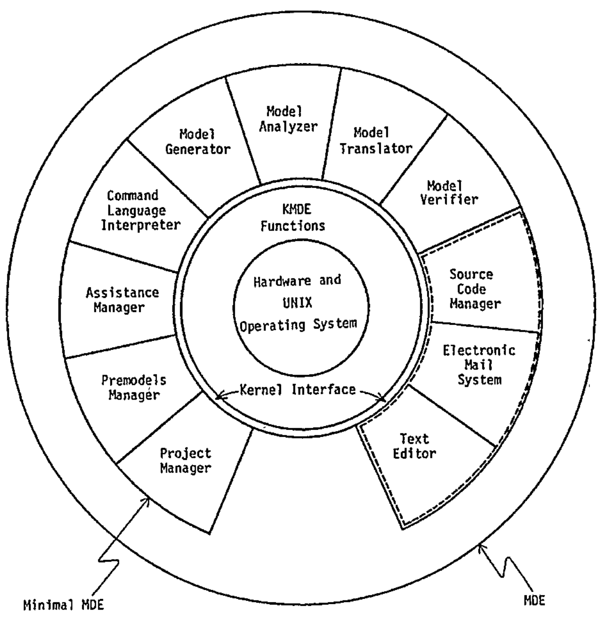

Figure 3. Layered Illustration of the Software Components of a Model Development Environment (MDE).

MODELING AND PROGRAMMING PARADIGMS

The tendency to view model development as identical with program development is generally criticized by those working in the modeling community. Differences between the two, especially when compared prior to 1980 say, are easily distinguished:

(1) Software systems implemented with "static" languages, such as FORTRAN, COBOL, and PL/I, represent design challenges far simpler than those faced by simulation modelers attempting to deal with the inherent complexity of representing dynamic dependencies.

(2) The validation of software systems in terms of conformance to design specifications can always be achieved by convincing the users that a change in the specification represents no loss in functionality; while the system being simulated can rarely be modified to conform to the behavior of the model.

(3) Software systems are rarely used by high-level managers, and when they are, in-depth understanding is unnecessary; while the use of a model relies almost totally on the degree to which credibility can be gained with the decision-making user.

The more recent emphasis on process-oriented software, abstract data types, concurrent programming, embedded systems, and object oriented languages appears to signal a shift so fundamental as to alter the software development paradigm. The consequence appears to be an evolution toward methodological approaches resembling the model based methodology.

Indications of this evolutionary trend are found in the paper by Balzer, et.al. [4], which calls for formal specifications developed and maintained by end users. The user becomes the systems analyst, or in the simulation domain, the modeler becomes the simulation programmer. Modifications are made at the specification level, and each respecification is subjected to a reimplementation, which is highly automated.

The concept of an automated programming assistant is not new; the concept is expressed by Winograd [44] over ten years ago, but the role of the assistant is expanded throughout the life cycle. The capabilities of the assistant are enumerated in a lengthy list ranging from project monitoring to program analysis. Clearly, such an automated assistant must possess or acquire considerable knowledge about the application domain. Domain dependency cannot be ignored on this level, a conclusion explained lucidly by Giddings [8] in a more recent paper that identifies the need for problem-solving environments and advances some crucial issues and implications related to the technology demanded for creating them.

Influenced by the early work of Zurcher and Randell [48], the view of Lehman [19], that the programming task is essentially a modeling task appears to be gaining acceptance. As a consequence, parallels between modeling and programming, modeling environments and programming environments, and the necessary requisites for both technologies are likely to be more broadly recognized. Contributions to the concepts of complexity and validation, such as found in [20] should prove valuable in both the modeling and programming camps.

SUMMARY

A brief chronology of simulation software helps to understand the current status, which finds simulation modeling in a transitional period. Viewed in the context of the model life cycle, the needs for more effective and efficient simulation model development can be identified. Some consensus is evident in the definition of tools, but the approaches to improvement are charted quite differently by researchers and practitioners in the simulation community. At this juncture no clear directions have been established. However, the indications of convergence in the software and model development paradigms can only be beneficial for both communities.

REFERENCES

[1] Advanced Research Projects Agency, "Requirements for ADA Programming Support Environments - STONEMAN," U.S. Dept. of Defense, Arlington, VA, 1980.

[2] Austell W.P., Jr., "Graphical Modeling and Simulation System (GMSS)," Simulation: Tools and Techniques Conference, Washington, DC.

[3] Balci, O., "Requirements For Model Development Environments," Technical Report CS83022-R, Department of Computer Science, Virginia Tech, Blacksburg, VA, 1983.

[4] Balzer, R., Cheatham, T.E., and Green, C., "Software Technology in the 1990's: Using a New Paradigm," *Computer, 16* (11), 1983, pp. 39-45.

[5] Buxton J.N. and Laski, J.G., "Control and Simulation Language," *Computer Journal, 5,* 1963, pp. 194-199.

[6] Clementson, A.T., "Extended Control and Simulation Language," University of Birmingham, England, 1973.

[7] Davies, N.R., "A Modular Interactive System for Discrete Event Simulation Modeling," *Proceedings of the Ninth Hawaii International Conference on System Sciences*, 1973, pp. 296.

[8] Giddings, R. V., "Accommodating Uncertainty in Software Design," *Communications ACM, 27* (5), 1984, pp. 428-434.

[9] Gordon, G., "The Development of the General Purpose Simulation System (GPSS)," In: *History of Programming Languages*, Wexelblat R (ed.), Academic Press, 1981, pp. 403-437.

[10] Handlykken P. and Nygaard, K., "The DELTA Description Language: Motivation, Main Concepts and Experience from Use," *Software Engineering Environments*, Hunke H (ed.), North Holland, 1981, pp. 157-172.

[11] Heidorn, G.E., "English as a Very High Level Language for Simulation Programming," *SIGPLAN Notices, 9(4)*, 1974, pp. 91-100.

[12] Heidorn, G.E., "Automatic Programming through Natural Language Dialogue: A Survey," *IBM J. Research and Development, 20*, 1976, pp. 302-313.

[13] Heimberger, D.A., "Interactive Modeling," Simulation and SIMSCRIPT Conference, Washington, DC, 1978.

[14] Holbaek-Hanssen, E., Handlykken, P., and Nygaard, K., "Systems Description and the DELTA Language," Report No. 4 (Publication No. 523), Norwegian Computing Center, Oslo, 1977.

[15] Kiviat, P.J., "GASP - A General Activity Simulation Program," Applied Research Laboratory, United States Steel Corporation, Monroeville, PA, 1963.

[16] Kiviat, P.J., "Digital Computer Simulation: Modeling Concepts," RAND Memorandum RM-5378-PR, Santa Monica, CA, 1967.

[17] Kleine, H., "A Vehicle for Developing Standards for Simulation Programming," *Proceedings of the Winter Simulation Conference*, 1977, pp. 730-741.

[18] Kleine, H., "Software Design and Documentation Language," JPL Publication 77-24, California Institute of Technology, 1977.

[19] Lehman, M.M., "Programs, Programming and the Software Life Cycle," Research Report No. 80/6, Department of Computing and Control, Imperial College, 1980, p. 48.

[20] Lehman, M.M., "Program Evolution," Research Report No. 82/1, Department of Computing and Control, Imperial College, 1982, p. 21.

[21] Markowitz, H.M., "Proposals for the Standardization of Status Description," Research Report RC 7782 (33671), IBM TJ Watson Research Center, Yorktown Heights, NY, 1979.

[22] Markowitz, H.M., Malhotra, A., and Pazel, D. P., "The EAS-E Application Development System: Principles and Language Summary," *Communications ACM, 27* (8), 1984, pp. 785-799.

[23] Mathewson, S.C., "Simulation Program Generators," *Simulation, 23(6)*, 1974, pp. 181-189.

[24] Mathewson, S.C., "Interactive Simulation Program Generators," *Proceedings of the European Computing Conference on Interactive Systems*, Brunel University, 1975, pp. 423-439.

[25] Mathewson, S.C., "Computer Aided Simulation Modeling and Experimentation," *Proceedings of the Eighth Australian Computer Conference*, 1978, pp. 9-13.

[26] Mathewson, S.C., "The Application of Program Generator Software and Its Extensions to Discrete Event Simulation Modeling," *IIE Transactions, 16* (1), 1984, pp. 3-18.

[27] Nance, R.E., "The Feasibility of and Methodology for Developing Federal Documentation Standards for Simulation Models," Final Report to the National Bureau of Standards, Department of Computer Science, Virginia Tech, 1977.

[28] Nance, R. E., "Model Representation in Discrete Event Simulation: The Conical Methodology," Technical Report CS81003-R, Department of Computer Science, Blacksburg, VA, 1981.

[29] Nance, R.E., "The Time and State Relationships in Simulation Modeling," *Communications ACM, 24(8)*, 1981, pp. 173-179.

[30] Nance, R.E., and Balci, O., "The Objectives and Requirements of Model Management," *Encylopedia of Systems and Control*, Pergamon Press, to appear, 1985.

[31] Nelson, S.S., and Lindstrom, G., "CONSIM: A Conversational Simulation Language Implemented through Interpretive Control Self-Modeling," Technical Report UUCS-77106, Department of Computer Science, University of Utah, 1977.

[32] Norden Division of United Aircraft Corporation, "Users Guide to NGPSS," Norden Report 4339R0003, 1971.

[33] Nygaard, K., and Dahl, O.J., "The Development of the SIMULA Languages," In: *History of Programming Languages*, Wexelblat R (ed.), Academic Press, 1981, pp. 439-493.

[34] Nygaard, K., and Handlykken, P., "The System Development Process -- Its Setting, Some Problems, and Need for Methods," *Software Engineering Environment*, Hunke H (ed.), North Holland, 1981, pp. 157-172.

[35] Oldfather, P.M., Ginsberg, A.S., and Markowitz, H.M., "Programming by Questionnaire: How to Construct a Program Generator," RAND Report RM-5129-PR, 1966.

[36] Oldfather, P., Ginsberg, A.S., Love, P.L., and Markowitz, H.M., "Programming by Questionnaire: The Job Shop Simulation Program Generator," RAND Report RM-5162-PR, 1967.

[37] Oren, T.I., A personal view on the future of simulation languages, *Proceedings of the UKSC Conference on Computer Simulation*, 1978, pp. 294-306.

[38] Oren, T.I., and Zeigler, B.P., "Concepts for Advanced Simulation Methodologies," *Simulation, 32(3)*, 1979, pp. 69-82.

[39] Pohoski, M.W., "A Top Level Description of the Ship Combat System Simulation," Naval Ocean Systems Center (jointly with NWC and NSWC), 1981.

[40] Teichroew, D., and Hershey, E.A., "PSL/PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing Systems," *IEEE Transactions on Software Engineering*, Vol. SE-3(1), 1977, pp. 41-48.

[41] Teichroew, D., Macasovic, P., Hershey, E.A., and Yamamoto, Y., Application of the entity-relationship approach to information processing systems modeling, In: *Entity-Relationship Approach to Systems Modeling and Design*, Chen, P. (ed.), North-Holland, 1980.

[42] Tocher, K.D., *The Art of Simulation*, Van Nostrand Company, Princeton, NJ, 1963.

[43] Tocher, K.D., and Owen, D.G., "The Automatic Programming of Simulations," *Proceedings of the Second International Conference on Operational Research*, 1960, pp. 50-68.

[44] Winograd, T., "Breaking the Complexity Barrier Again," *SIGIR Forum: Proceedings of the SIGPLAN-SIGIR Interface Meeting, IX* (3), 1984, pp. 13-22.

[45] Zeigler, B.P., *Theory of modelling and simulation*, John Wiley and Sons, 1976.

[46] Zeigler, B.P., "Concepts and Software for Advanced Simulation Methodologies," *Simulation with Discrete Models: A State-of-the-Art View*, Oren, T.I., Shub, C.M., and Roth, P.F. (eds.), IEEE, 1980, pp. 25-44.

[47] Zeigler, B.P., "System-Theoretic Representation of Simulation Models," *IIE Transactions, 16* (1), 1984, pp. 19-34.

[48] Zurcher, F. W., and Randell, B., "Iterative Multi-Level Modelling--A Methodology for Computer System Design," *Proceedings of the IFIPS Congress*, Edinburgh, 1968, pp. 867-871.