# From Discrete Event Simulation to Discrete Event Specified Systems (DEVS)

Bernard P Zeigler, Alexander Muzy

# From Discrete Event Simulation to Discrete Event Specified Systems (DEVS)

Bernard P. Zeigler* and Alexander Muzy**

*Arizona Centre for Integrative Modeling and Simulation and RTSync, Potomac, MD 20854, USA.
e-mail: zeigler@rtsync.com
**CNRS, I3S, Université Côte d'Azur, 06900 Sophia Antipolis, France.
email: alexandre.muzy@cnrs.fr

**Abstract:** In this presentation we discuss the evolution of simulation from its origin in design of computer and communication systems based on event routines, to the conceptualization of the objects under study as systems, to the behavior generation of DEVS models representing a wide variety of cyber-physical forms. Indeed, the history of computer simulation programming languages parallels, and is intertwined with, the evolution of programming concepts from hardware-dependent binary scripts to successively more abstract model-based generic frameworks.

***Keywords:*** Discrete Event Dynamic Systems, Discrete Event System Specification, Systems theory.

## 1. INTRODUCTION

It is some sixty years after the first use of a form of digital simulation appeared that we roughly identify as event-oriented simulation. At its advent, it was mainly thought to be distinct from classical simulation in being a form of programming associated with the recent introduction of the digital computer and applied to operational research problems. In contrast, classical simulation was taken to be a form of numerical solution applicable to physics and related sciences whose speed could be greatly increased with mechanical rather than hand calculation. The distinctive trend of computational science is the continual trend toward greater abstraction and identification of the true underlying commonalities and distinctions between apparently different phenomena. This trend is realized in the evolution of concepts relating to event-oriented simulation that transpired since its inception and which forms the core of the historical perspective on discrete event simulation that we present here.

The abstraction we adopt in the session presentation was first introduced a decade after event-oriented simulation. We will elaborate on the concept of "system" as defined by Wymore[1] (1967) as a basis for unifying various forms of discrete and continuous simulation. Briefly, *Theory of Modeling and Simulation* (1976) defined the Discrete Event System Specification (DEVS) formalism as a specification for a subclass of Wymore systems that could capture all the relevant features of the *models* underlying event-oriented simulations; In contrast, Discrete Time Systems Specification (DTSS) and Differential Equation System Specification (DESS) specify distinct *subclasses* of Wymore systems – the first, as a basis for discrete time models (including those specified by finite automata and cellular automata); the second to represent the continuous models underlying classical numerical solvers.

1 Others like Zadeh (Zadeh and Desoer, 1963), Klir (Klir and Elias, 2002), and Arbib (1970) had similar definitions.

## 2. EARLY HISTORY

While K.D. Tocher was not the first to practice event-oriented simulation, he appears to be the first to conceive discrete events *as* the right abstraction to characterize the models underlying the techniques that he and others were adopting in the mid-1950s. According to Hollocks (2008), Tocher's core idea was of a system consisting of individual components, or 'machines', progressing as time unfolds through 'states' *that change only at discrete 'events'*. Indeed, DEVS took this idea one step further using the set theory of logicians and mathematicians [(Whitehead and Russel, 1910), Bourbaki (1930)] and its use by Wymore (1967).

In the presentation, we summarize the history of discrete event simulation (now using Tocher's characterization) beginning with Monte Carlo sampling and the limitations of analytic queueing analysis. These forerunners lead to machine language coding, attempts to handle specific real systems, and eventually to general purpose tools: General Simulation Program (GSP) (Reitment, 1988) (later General Purpose Systems Simulator (GPSS) (Schriber 1974)), A General Activity Simulation Program (GASP) (Kiviat, 1963) and SIMSCRIPT (Markowitz et al., 1963) in the USA, Simple universal language (SIMULA) (Dahl and Nygaard, 1967) in Norway, and GSP (Tocher, 1967) and Control and Simulation Language (CSL) (Buxton and Laski, 1969) in the United Kingdom. The modeling strategies behind these programming languages became encapsulated in the concept of world views: event scheduling, activity scanning, and process-interaction (Lackner, 1964). Zeigler (1984) characterized these world views showing that they could all

be represented as subclasses of DEVS thus also suggesting its universality for discrete event models including those or other representations such as Timed Automata and Petri Nets. Also at the same time the distinction between modular DEVS (components interacting through their inputs/outputs) and non-modular (components being able to change others state in a non-encapsulated way) DEVS was made showing that the world views all fit within the non-modular category. Moreover, while the modular class was shown to be equivalent to that of the non-modular one, it better supported the impending concepts of modularity, object orientation, and distributed processing on the software engineering horizon.

## 3. FRAMEWORK FOR SIMULATION AND DEVS

### *3.1 Modeling and Simulation Framework (MSF)*

The modeling and simulation framework (MSF) presents entities and relationships of a model and its simulation. The MSF separates models from simulators as entities that can be conceptually manipulated independently and then combined in a relation which defines correct simulation. The Experimental Frame defines a particular experimentation process, e.g., Latin hypercube sampling for yielding model outcome measurements in accordance with specific analysis objectives. The MSF helps clarify many of the issues involved in such activities. Mismatch between the simulation's time management policy and the model's time advance approach creates significant errors (such as in temporal ordering of events) in even the simplest Modeling and Simulation (M&S). The MSF underlies the DEVS Simulation Protocol which **provides provably correct simulation execution of DEVS models** thereby obviating the errors well as throwing light on the source of such conflicts in non-DEVS compliant simulations.

### *3.2 DEVS Formalism*

A DEVS model is a system-theoretic concept specifying inputs, states, outputs, similar to a state machine (Mittal and Risco Martin, 2013). Critically different however, is that it includes a time-advance function that enables it to represent discrete event systems, as well as hybrids with continuous components (Nutaro, 2011) in a straightforward platform-neutral manner. In the presentation we will discuss these points:

- DEVS formalizes what a model is, what it must contain, and what it doesn't contain (experimentation and simulation control parameters are not contained in the model).
- DEVS is universal and unique for discrete event system models: any system that accepts events as inputs over time and generates events as outputs over time is equivalent to a DEVS: its behavior and structure can be described by such a DEVS.
- DEVS-compliant simulators execute DEVS models correctly, repeatably, and efficiently. Closure under coupling guarantees correctness in hierarchical composition of components.
- DEVS models can be simulated on multiple different execution platforms, including those on desktops (for development) and those on high-performance platforms, such as multi-core processors (Muzy et al., 2016).

### *3.3 DEVS Simulation Protocol*

The DEVS Simulation Protocol (Zeigler and Sarjoughian, 2013) is a general distributed simulation protocol that prescribes specific mechanisms for:

- declaring the participants in the simulation (component models = federates)
- declaring how federates exchange data
- executing an iterative cycle that
  - controls how time advances (time management)
  - determines when federates exchange messages (data exchange management)
  - determines when federates do internal state updating (state update management)

The protocol guarantees correct simulation in the sense that, if the federates are DEVS models then the federation is also a well-defined DEVS coupled model. Distinct from the High Level Architecture (an IEEE standard for distributed simulation), the DEVS protocol prescribes specific *time, data exchange, and state update* management processes. Moreover, hese benefits do not imply undue performance degradation. The Parallel DEVS Simulation Protocol provides close to the best possible performance except possibly where activity is very low or coupling among components is very small (Zeigler and Nutaro, 2015). There are numerous implementations of DEVS simulators (see the list by Wainer, 2012). In particular ADEVS (Nutaro 2008, 2011) is also distinguished by its support for both discrete event and continuous dynamic systems, both of which are simulated within the DEVS framework (Nutaro, 2014), (Nutaro and Sarjoughian, 2014). This gives it the capability to simulate, within the MSF, the interactions of sub-systems characterized by discrete event dynamics (e.g., communication networks and command as well as control systems) and continuous, physical dynamics (e.g., the trajectories of ballistic missiles and their interceptors).

## 4. TIMELINE HISTORY OF DEVS DEVLOPMENTS

A brief historical retrospective, time-sequenced in Figure 1, is presented on developments including refinements,

elaborations, and extensions of the DEVS formalism with associated references.

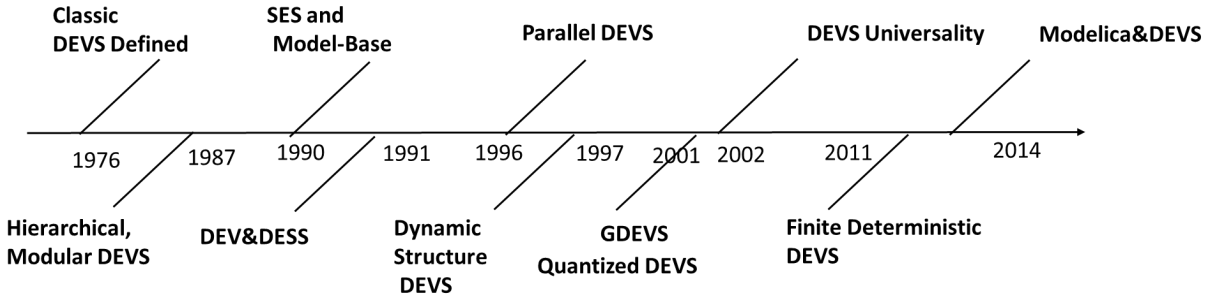


Figure 1. Timeline for DEVS Developments

DEVS as a formalism for simulation-related modeling was defined by Zeigler in the late 1960s and tied to the emerging system theoretic ideas of that period especially to the formulations of (Wymore, 1967) but also to those of (Mesarovic, 1964) and (Arbib, 1970). We sometimes refer to that original formulation as Classic DEVS in contradistinction to the later reformulation known as Parallel DEVS twenty years later. Parallel DEVS removed the requirement for sequential execution of events, allowing simultaneous sending and receiving of inputs and outputs among components. Accordingly the basic atomic model of Classic DEVS was extended to enable the modeler to determine how the model reacts to collections of multiple inputs as well as other concomitant extensions of the basic structure. In the period between the appearances of Classic and Parallel DEVS certain inherent features of DEVS were elaborated in the literature such as: a) the modularity enabled by its incorporation of external event inputs, b) the hierarchical construction enabled by its closure under coupling property that is inherited from systems concepts, and c) its connection to the System Entity Structure. The latter represents an excursion beyond classical systems concepts with an ontological mean of expressing model compositions and their alternatives that can be pruned to meet simulation objectives. Also in the 1990s the general system theoretic basis of DEVS and its formulation as a co-equal system specification with differential equation systems was explicitly demonstrated with the definition of the DEV&DESS specification (Praehofer, 1991) that formalizes the hybrid combination of discrete events and differential equations. Allowing models to change their internal structure in DEVS was an important extension greatly opening up the realm of possibilities for modeling of modern technological systems. The ability to express computational approaches to differential equation systems was instantiated by the Quantized State Systems in DEVS (Kofman and Junco, 2001) and Generalized DEVS (Giambiasi et al., 2001) extensions. These developments and others were systematically elucidated in the second edition of Theory of Modeling and Simulation in 2000 (Zeigler et al., 2000). This volume has been instrumental in stimulating DEVS-based research and developments throughout the world and in many disciplines as evidenced in the over 5000 citations it has received. A few of the subsequent developments in theory (proof of DEVS universality) and practice (the combination of DEVS and Modelica software packages for industrial strength simulation) are suggested in Figure 1.

## 5. RELATION TO OTHER DEDS FORMALISMS

The universality of DEVS representation implies that arbitrary discrete event dynamic systems, such as expressed as Timed Automata (Dacharry and Giambiasi, 2008) or Petri Nets (Vangheluwe, 2000), can be advantageously represented as DEVS models. Furthermore, such DEVS models provide a basis for the design of event-based logic control showing how classical process control can be readily interfaced with rule-based symbolic reasoning systems (Zeigler, 1989).

## 6. DEVS TODAY

A representative sample of current research and development in DEVS is listed below (Bisgambiglia, 2016):

- **Theory**
  - Extension of the System Entity Structure for Hierarchical Abstraction
  - Formal Model-Checking Methods and DEVS
  - DEVS M&S of Multilayered Social Networks
  - Multicomponent Formulation of DEVS
  - Verification and Validation (Zeigler and Nutaro, 2016)
- **Applications**
  - Parallel Simulation of DEVS Spiking Neurons
  - DEVS Smart Phone Simulation
  - DEVS Encoding into Timed Petri Nets for Hardware Implementation
  - Multicomponent DEVS for Multi-agent Systems
- **Simulation and Optimization**
  - DEVS Architecture for Simulation-based Optimization
  - Optimization of DEVS Distributed Simulations
  - Reproducibility of High Performance Stochastic Simulations

- **DEVS Development Environments**
  - VLE-Virtual Lab Environment[2]
  - MS4 Me[3]
  - DEVSimPy[4]
  - GRADES (Graph-based and RAndom DiscrEte-event Simulator)[5]
  - PythonDEVS[6]
  - ProDEVS: Petri Net encoding of DEVS (Vu et al., 2015)
  - fwkDEVS: OO-Framework for DEVS (Bisgambiglia. 2016)

## 7. THEORY DEVELOPMENT FUTURE PROGNOSTICATION

Finally, we mention particular research relating to DEVS that aims at integrating basic systems foundations (with iterative specification) and adding Dynamic Structure Systems (Muzy and Zeigler, 2014), Markov and probabilistic systems modeling in DEVS (Zeigler et al., 2016) and pseudorandom simulations (Muzy et al., 2016).

## 8. ACKNOWLEDGEMENT

The authored gratefully acknowledge the help received from Gwynn Thayer and Jennifer Baker of the North Carolina State University Libraries, Special Collections Research Center for their indispensable help in searching the audio scripts of the Computer Simulation Archive for the origins of the term “discrete event”.

## REFERENCES

Aribib, M.A, (1970). Theories of Abstract Automata, McGraw Hill.

Bisgambiglia. P.A. (2016). Les journées DEVS francophones : Théorie et Applications / Workshop http://www.reseau-devs.org/jdf-2016.

Bourbaki, N. (1930). Wikipedia, https://en.wikipedia.org/wiki/Nicolas_Bourbaki.

Buxton JN and Laski JG (1969). Control and simulation language. Comput J 5: 194–199.

Chow, A.(1996). Parallel DEVS: A Parallel, Hierarchical Modular Modeling Formalism and its Distributed Simulator, Transactions of the SCS, Vol 13, #2, pp. 55-68.

Dacharry, H., & Giambiasi, N. (2008). *DEVS and timed automata for the design of control systems*, Nova Science Publishers, 193-222.

Fujimoto, R.M. (1999). Parallel and Distribution Simulation Systems (1st ed.). John Wiley & Sons, Inc.

Garzia, R.F. , Garzia, M.R. and Zeigler, B.P. (1986). Discrete event simulation", IEEE Spectrum, pp. 32-36,.

Giambiasi, N., Escude, B. and Ghosh, S. (2001). GDEVS: A Generalized Discrete Event Specification for Accurate Modeling of Dynamic Systems. ISADS 2001: pp. 464-469a

Hollocks, B.W. (2008). Intelligence, innovation and integrity-KD Tocher and the dawn of simulation, available at:

http://www.themedfomscu.org/media/elip/jos200815a.pdf.

Hwang, M.H. (2001). Taxonomy of DEVS subclasses for standardization, TMS-DEVS pp. 152-159,

Kiviat P.J. (1963). GASP—A General Activity Simulation Program. Applied Research Laboratory, US Steel Corporation, Monroeville, PA.

Klir, G. J. and Elias, D. (2002). Architecture of Systems Problem Solving (2 ed.). Da Capo Press, Inc.

Kofman, E. and Junco, S. (2001). Quantized-state systems: a DEVS Approach for continuous system simulation, Transactions of The SCS,Volume 18, # 1, pp. 2-8.

Lackner, M. R. (1964). Digital Simulation and System Theory, (SDC Document No. SP-1612), System Development Corp. Santa Monica, Calif., April 6.

Markowitz H.M., Hausner B. and Karr H.W. (1962). Simscript: The simulation programming language. Rand Corporation Report, Rm-3310, Cambridge, MA.

Mesarovic, M. D. (1964). Foundations for a general systems theory. *Views on General Systems Theory*, 1-24.

Mital, S., Risco Martin, J. L. (2013). Netcentric System of Systems Engineering with DEVS Unified Process, CRC Press.

Muzy, A. and Zeigler, B.P. (2014). Specification of dynamic structure discrete event systems using single point encapsulated control functions, International Journal of Modeling, Simulation, and Scientific Computing (IJMSSC) Vol. 5, Issue 3, 1450012.

Muzy A., Lerasle M., Grammont F., Dao V.T., Hill D.R.C. (2016). Parallel and pseudorandom discrete event system specification vs. networks of spiking neurons: Formalization and preliminary implementation results, accepted for publication in International Conference on High Performance Computing & Simulation (HPCS) conference.

2 Available at: http://www.vle-project.org/, last connection: 02/11/2016.
3 Available at: http://ms4systems.com, last connection: 02/11/2016.
4 Available at: http://devsimpy.univ-corse.fr/attachment/481281/, last connection: 02/11/2016.
5 Available at: https://redmine.i3s.unice.fr/projects/compsys, last connection: 02/11/2016.
6 Available at: http://msdl.cs.mcgill.ca/projects/projects/DEVS/, last connection: 02/11/2016.

Nance R.E. (1996). A history of discrete-event simulation programming languages. In: Bergin TJ and Gibson RJ (eds). History of Programming Languages Vol. II ACM Press and AddisonWesley Publishing Company: New York, pp 369–427.

Nutaro, J. (2008). On constructing optimistic simulation algorithms for the discrete event system specification. ACM Transactions on Modeling and Computer Simulation, 19(1), pp. 1-21.

Nutaro, J. (2011). Building Software for Simulation: Theory and Algorithms with applications in C++. Wiley.

Nutaro, J. (2014). An extension of the OpenModelica compiler for using Modelica models in a discrete event simulation, SIMULATION, December; vol. 90, 12: pp. 1328-1345.,

Nutaro, J. and Sarjoughian, H. S. (2004). Design of Distributed Simulation Environments: A Unified System-Theoretic and Logical Processes Approach. SIMULATION 80(11), pp 577-589.

Nutaro, J., Kuruganti, P. T., Protopopescu, V. and Shankarm M. (2012). The split system approach to managing time in simulations of hybrid systems having continuous and discrete event components. SIMULATION 88(3), pp. 281-298.

Nygaard K. and Dahl O. (1978). The development of the SIMULA languages. ACM SIGPLAN Notices 13: 245–272.

Praehofer, H. (1991). System Theoretic Formalisms for Combined Discrete-Continuous System Simulation. Int. J. Gen. Sys., 19(3): p. 219-240.

Reitman J. (1988). A concise history of the ups and downs of simulation. In: Abrams M, Haigh P and Comfort J (eds). Proc. of the 1988 Winter Simulation Conference—San Diego. IEEE: Piscataway, NJ, pp 1–6.

Shaffer, A.R. (2012). The Value of Modeling and Simulation for the Department of Defense M&S Journal, pp 2-3

Schriber, T, (1974). Simulation using GPSS. Wiley. ISBN 9780471763109.

Tocher KD (1967). PLUS/GSP III Specification. United SteelCompanies Ltd, Department of Operational Research, Sheffield.

Steiniger, A. and Uhrmacher, A.M. (2016). Intensional Couplings in Variable-Structure Models: An Exploration Based on Multilevel-DEVS. ACM Trans. Model. Comput. Simul. 26, 2, 27 pages. DOI=http://dx.doi.org/10.1145/2818641.

Vangheluwe, H. L. (2000). DEVS as a common denominator for multi-formalism hybrid systems modelling. In *Computer-Aided Control System Design, 2000. CACSD 2000. IEEE International Symposium on Computer-Aided Control System Design*, 129-134.

Vu, L. H., Foures, D., & Albert, V. (2015). ProDEVS: an Event-Driven Modeling and Simulation Tool for Hybrid Systems using State Machines. In *Simutools 2015 Eighth EAI International Conference on Simulation Tools and Techniques*, Brussels, Belgium, No. 8, 29-37.

Wainer, G.A. (2012). Available at: http://www.sce.carleton.ca/faculty/wainer/standard/tools.htm

WG - HLA Evolved Working Group (2010). Modeling and Simulation High Level Architecture, IEEE 1516–2010, The Institute of Electrical and Electronics Engineers, .

Whitehead, A.N. and Russell, B. (1910). Principia mathematica 1 (1 ed.), Cambridge: Cambridge University Press, JFM 41.0083.02

Wymore, A.W. (1967). Mathematical Theory of Systems Engineering: The Elements, Wiley/

Zadeh, L.A. and Desoer, C.A. (1963). Linear System Theory, The State Space Approach, MrGraw Hill.

Zeigler, B.P. and Sarjoughian, H.S. (2013). Guide to Modeling and Simulation of Systems of Systems Springer.

Zeigler, B.P., Praehofer, H., & Kim, T.G. (2000). Theory of Modeling and Simulation (2nd ed.). Academic Press.

Zeigler, B.P. (1987). Hierarchical, Modular Discrete Event Models in an Object Oriented Environment, Simulation J., vol. 49, no. 5, pp. 219-230.

Zeigler, B.P. (1976). Theory of Modelling and Simulation, Wiley.

Zeigler, B.P. (1984). Multifaceted Modelling and Discrete Event Simulation, Academic Press.

Zeigler, B.P. (1984). System-theoretic representation of simulation models, IIE Transactions, pp. 19-34.

Zeigler, B.P. (1989). DEVS representation of dynamical systems: EventBased Intelligent Control, Proc. of the IEEE, vol. 77, no. 1, pp. 72-80.

Zeigler, B. P. and Sarjoughian, H. S. (2013). DEVS Simulation Protocol. In *Guide to Modeling and Simulation of Systems of Systems*, Springer London, pp. 105-124.

Zeigler, B.P. Nutaro J. and Seo, C. (2015). What's the Best Possible Speedup Achievable in Distributed Simulation: Amdahl's Law Reconstructed, DEVS TMS, SpringSim.

Zeigler, B.P. and Nutaro J. (2016). Towards a Framework for More Robust Validation and Verification of Simulation Models for Systems of Systems, *Journal of Defense Modeling and Simulation: Applications, Methodology, Technology, Vol. 13(1) 3–16 2015 DOI: 10.1177/1548512914568657.*

Zeigler, B.P. Nutaro J. and Seo, C. (2016). Combining DEVS and Model-Checking: Concepts and Tools for Integrating Simulation and Analysis, to appear in *Int. J. Process Modeling and Simulation*. Special Issue on: "New Advances in Simulation and Process Modelling: Integrating New Technologies and Methodologies to Enlarge Simulation Capabilities".