A survey of modelling and simulation software frameworks using Discrete Event System Specification

Romain Franceschini[1], Paul-Antoine Bisgambiglia[1], Luc Touraille[2], Paul Bisgambiglia[1], and David Hill[2]

1 **University of Corsica Computing Laboratory**
UMR SPE 6134 CNRS, UMS Stella Mare 3460, Campus Grimaldi, 20250 Corti
`{r.franceschini,bisgambiglia,bisgambi}@univ-corse.fr`

2 **CNRS, UMR 6158, ISIMA LIMOS, Blaise Pascal University**
BP 10448, F-63000 Clermont-Ferrand, France
`{touraille@isima.fr, david.hill}@univ-bpclermont.fr`

Abstract

Discrete Event System Specification is an extension of the Moore machine formalism which is used for modelling and analyzing general systems. This hierarchical and modular formalism is time event based and is able to represent any continuous, discrete or combined discrete and continuous systems. Since its introduction by B.P. Zeigler at the beginning of the eighties, most general modelling formalisms able to represent dynamic systems have been subsumed by DEVS. Meanwhile, the modelling and simulation (M&S) community has introduced various software frameworks supporting DEVS-based simulation analysis capability. DEVS has been used in many application domains and this paper will present a technical survey of the major DEVS implementations and software frameworks. We introduce a set of criteria in order to highlight the main features of each software tool, then we propose a table and discussion enabling a fast comparison of the presented frameworks.

1998 ACM Subject Classification I.6 Simulation and modeling

Keywords and phrases DEVS, Framework, Survey, Modelling, Simulation

Digital Object Identifier 10.4230/OASIcs.ICCSW.2014.40

1 Introduction

Simulation has become a popular tool to study a broad range of systems. The growing number and quality of simulation software requires expertise for their evaluation. Selecting an appropriate framework is an important issue to simulation practitioners.

Our study will be restricted to software based on discrete event systems (DES). A Discrete Event Sytem is a discrete-state, event driven dynamical system in which the state space is described by a discrete set, and states evolve in terms of asynchronous occurrence of discrete events over time. In DES theory, states, events and transition functions are defined by a five-tuple [6]: $M = \langle S, s_0, \lambda, \delta, \Sigma \rangle$ where: S is the set of states; s_0 is the initial state vector; Σ is the set of events; $\delta : S \times \Sigma \to S$ the state transition function; $\lambda : S \times \Sigma \to \Sigma_d$ the output function, where Σ_d and Σ_{ud} the set of detectable and undetectable events, respectively $\Sigma = \Sigma_d \cup \Sigma_{ud}$. Actually, DES represent many technological and engineering systems such as communication networks, computer networks, manufacturing systems, transportation systems, natural systems, and more. So far, many modelling approaches of DESs have been proposed and developed, notably including finite automata, fuzzy automata, Petri nets and the DEVS formalism.

The DEVS formalism [4, 36] is considered as universal for discrete event dynamic systems and is capable of representing a wide class of other dynamic systems. DEVS can simulate discrete time systems such as cellular automata and can also approximate, as closely as desired, differential equation systems, continuous systems, etc. DEVS is a modular formalism which permits the modelling of causal and deterministic systems. A DEVS atomic (behavioral) model is described by the following formula: $M = \langle X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta\rangle$, where X the list of the model inputs; Y the list of the model outputs; S the system states; ta the time advance function; $\delta_{ext} : S \times X \to S$ is the external transition function; $\delta_{int} : S \to S$ is the internal transition function and $\lambda : S \to S$ is the output function. A complex system can be designed as a coupling of several simpler systems using a DEVS coupled (structural) model, described by the following formula: $MC = \langle X, Y, D, \{M_d \mid d \in D\}, EIC, EOC, IC, L\rangle$, with X being the input ports and values set; Y the output ports and values set; D the set of components; EIC the total set of input couplings ($X \to d$), which links the coupled model to its components; EOC the total set of output couplings ($d \to Y$), which links components to the coupled model; IC the total set of internal couplings, which links components with each other; L a list of priorities among components. To define the simulation semantics of DEVS models, Zeigler introduced abstract simulators. The advantage with such an approach lies in the separation between the models and the simulators. DEVS provides the algorithms used to simulate a model hierarchy, introduced in [36].

Selecting a software framework that meets the requirements needs a full examination of many factors and it is a difficult task. The aim of this paper is to propose a survey of research done in the field of M&S. The review is limited and the suggestion criteria are defined to assist decision makers in evaluating and selecting software. In this paper we will provide a broad comparison, and propose a benchmark on a subset of three tools. Section 2 describes our comparative criteria (programming language, version, GUI, DEVS extensions...). Section 3 lists the studied software [1, 5, 8, 9, 11, 14, 16, 20, 22, 26, 28, 30, 33, 34, 38]. Before concluding, Section 4 provides a comparison grid according to our selected criteria and a discussion presenting the comparison of DEVS software frameworks.

2 Discussion on criteria

Before any evaluations, it is essential to establish criteria of comparisons. Evaluating and selecting software frameworks that meet users requirements is a difficult process. The aim is to provide a basis to improve the process of selection of the software frameworks [13, 19]. Several methods have emerged to formulate a software evaluation process. Many papers have proposed their list of criteria with the lack of a standard common list. A standard list of criteria, an explanation and an example for each criterion could overcome some pitfalls. Due to progress in the field, new computer technology and software updates, it may not be possible to provide a standard list of criteria. This paper focus on 8 sets of criteria, chosen to give the best possible description framework. The latter were defined according to our application domain, i.e. the academic domain: **1**) Software version. If it is no longer updated or maintained it can become deprecated, as JDEVS [8]; **2**) The programming language is a very important criterion, especially if we want to make performance comparisons. This type of comparison has already been proposed [29]. Language popularity and libraries available are also important aspects to consider; **3**) The quality of documentation; **4**) Simulator algorithms and the proposed extensions: for 30 years the formalism has evolved, a parallel version of the simulator has been proposed [7], as numerous extensions for real-time systems, continuous, cellular [1, 31]; **5**) Model sets. DEVS formalism has been highlighted for its modular side. A framework should provide a set of models ready to be used; **6**) Network management. This is the opportunity to run models remotely using a web service [2, 17] or to provide models through middleware (HLA, SOA) [18, 32, 35]; **7**) Design tools. Like DEVS, these frameworks have evolved. A lot of frameworks are helping modelers with a GUI. Today, with the contributions of model driven engineering (MDE), very powerful tools (meta-modelling) are used to generate the model code, to mark them interoperable and to standardise simulators [10, 16, 25]; and **8**) Analysis and interpretation tools through visualization, statistics and reports. The next section will describe the major DEVS frameworks.

3 Framework description

In this section we study a non-exhaustive list of several DEVS-based simulators by giving their description.

CD++ [30] is a DEVS M&S toolkit that provides a library of C++ classes to specify models in several DEVS formalisms and simulate them. The supported formalisms include classical DEVS (CDEVS) and parallel DEVS (PDEVS), but the focus is on Cell-DEVS [31], an extension integrating cellular automata and DEVS. CD++ can handle in a single simulation several types of models, e.g. parallel DEVS and Cell-DEVS models. Simulations can be performed either locally or remotely, by sending model specifications to a simulation server. Depending on their types, CD++ models are specified either as C++ classes or through text files following a custom format. In addition to the simulation kernel, CD++ provides a plugin allowing edition of models both textually and graphically, as well as visualization of simulation results.

DEVSJava [23] is a Java library for modelling and simulating PDEVS, Dynamic-Structure DEVS and Real-Time-DEVS models. It is co-directed by B.P. Zeigler, H. Sarjoughian and R. Lysecky. DEVSJava provides a set of custom container classes for storing entities manipulated by models. These containers are used to develop DEVS models according to the class hierarchy defined by the library. Models are specified as Java classes and can then be simulated with the simulation processors implemented in the library. Local simulation, distributed simulation and real-time simulation are supported. Later versions of DEVSJava include a dynamic-structure modelling feature. The DEVSJava library is now included in a larger software suite for M&S called DEVS-Suite, which provides some graphical facilities for editing models, controlling simulation and visualising results [15]. Subsequently B.P. Zeigler created a company and incorporated some of DEVSJava and DEVS-Suite tools in MS4ME [33].

James II [12, 38] is a generic M&S platform, written in Java, which is developed under the coordination of A.M. Uhrmacher. Its aim is to provide an extensible platform that can integrate any modelling formalism, simulation algorithms, and tools. To do so, it uses a flexible plugin system that makes the addition of new features in the platform quite seamless. The architecture of James II focuses on minimizing coupling between modules. The core of the platform provides a set of services and classes for use by other packages, such as random number generation, data structures, mathematical functions, serialization, etc.). It also handles the graphical user interface and the plugin system. Other features are implemented as plugins. The most important types of plugins are modelling formalisms, simulation algorithms, editors and visualizers, but other plugin types can be defined. The last version at the time of this writing (v0.9.6) comes bundled with several formalisms, among others DEVS, PDEVS, PdynDEVS and cellular automata, along with various simulation algorithms (sequential, multi-threaded, etc.), editors and visualizers for each.

Virtual Laboratory Environment (VLE) [21] is an M&S platform based on PDEVS, written in C++ and mainly developed by Gauthier Quesnel. VLE is now integrated in the RECORD platform supported by the Applied Mathematics and Computer Science department of INRA (named MIA). Like James II, its architecture is quite modular to facilitate the addition of new features. The core of VLE consists in a set of class libraries (VFL), that implement several formalisms (Petri nets, 2D/3D cellular automata, Quantized State Systems (QSS), etc.). VLE provides a PDEVS simulator, different ways to observe graphically the models in real time, through the Eyes of VLE (EOV), and a graphical user interface (GVLE) for editing models and their couplings/hierarchy.

PyDEVS & PyPDEVS PyDEVS is a DEVS Modelling and Simulation Package implemented in Python [3] and mainly developed by The Modelling, Simulation and Design lab (MSDL) headed by Prof. Hans Vangheluwe. The package provides an easy way to model and simulate hierarchical DEVS. It is based on classic DEVS formalism. Newer version of the package called PyPDEVS [24] is available with PDEVS implementation and a distributed simulations feature. The package is also used as a basis for two other frameworks: AToM^3 [16], a multi-paradigm modelling tool with meta-modelling and model-transforming features that relies on model driven engineering (MDE) concepts and DEVSimPy [5], on open source project supported by University of Corsica that provides a GUI to facilitate both the coupling and reusability of models.

PowerDEVS is a software tool for classical DEVS M&S oriented toward the simulation of hybrid systems [1]. Developed by E. Kofman team of Rosario University, it allows defining atomic DEVS models in C++ which can be graphically coupled and translated in C++ code. It gives the possibility to perform simulations in real time, allowing the design and automatic implementation of digital controllers. It can be interconnected with Scilab.

SimStudio [27] is an architecture built upon the DEVS formalism that aims at integrating tools for M&S, analysis and collaboration through Model-Driven Engineering (MDE) features such as code generation.

DEVS-Ruby [9] is a library that allows formal specifications of CDEVS and PDEVS models. It provides an internal Domain-Specific Language (DSL) which can be extended to meet domain specific vocabulary of modelers.

4 Comparison and discussion

This section provides a comparison of the software listed in section 3 according to our selected criteria (see section 2) as long with a performance analysis of some frameworks.

4.1 Comparison

The aim of this comparison is to guide, according to its needs, any potential newcomer wishing to use a DEVS framework. As shown in Table 1, most of listed software are still maintained, except for SimStudio and AToM^3. Note that MS4Me is proprietary and that CD++ is not available to download except if you have a user access to the corresponding wiki. There is not a wide variety of languages used (C++, Java, Python, Ruby), but they are fundamentally various in terms of characteristics. We rate documentation from 1 to 5 based on code documentation, examples, tutorials, wikis of each simulator and provide a link to a website if available. Table 1 lists all approached frameworks, but subsequent

Table 1 Framework comparison grid based on criteria 1, 2 and 3.

DEVS Frameworks		**Criteria**			
Name	**Ref.**	**1**	**2**	**3**	
		Version	**Lang.**	**Documentation**	
aDEVS	[20]	2014	C++	**2**	`http://web.ornl.gov/~1qn/adevs/`
CD++	[30]	2013	C++	**5**	`http://cell-devs.sce.carleton.ca/`
PowerDEVS	[1]	2014	C++	**4**	`http://sourceforge.net/projects/powerdevs/files/`
SimStudio	[27]	2010	C++	**1**	
AToM^3	[16]	2006	Python	**4**	`http://atom3.cs.mcgill.ca/`
MS4Me	[37]		Java	**5**	`http://www.ms4systems.com/pages/main.php`
JAMES II	[38]	2014	Java	**5**	`http://wwwmosi.informatik.uni-rostock.de/mosi/projects/cosa/james-ii/`
VLE	[22]	2014	C++	**5**	`http://www.vle-project.org/wiki/Main_Page`
PyDEVS	[3]	2014	Python	**4**	`http://msdl.cs.mcgill.ca/projects/projects/DEVS/`
DEVSimPy	[5]	2014	Python	**3**	`http://devsimpy.univ-corse.fr/`
PyPDEVS	[24]	2014	Python	**4**	`http://msdl.cs.mcgill.ca/people/yentl/50_master`
DEVS-Ruby	[9]	2014	Ruby	**3**	`http://devs-ruby.github.io/devs_ruby/`

Table 2 Framework comparison grid based on 4th criterium.

DEVS Frameworks		**Criteria**		
Name	**Ref.**	**4**		
		Simulator Algorithms	**Extensions Concepts**	**Systems modelling**
aDEVS	[20]	PDEVS	Cell Space, dynDEVS	social, ecological, computer networks and computer architecture, military
CD++	[30]	CDEVS, PDEVS, distributed	Cell-DEVS, RT-DEVS, E-CD++, DEVStone	social, ecological, computer networks, environmental
PowerDEVS	[1]	CDEVS	QSS, QSS2, RT-DEVS	hybrid, physical, Scilab interconnexion
JAMES II	[38]	PDEVS, distributed	FDDEVS, SoS, components, agent	hybrid, space
VLE	[22]	PDEVS	DSDE	agent, agro-ecosystems
PyPDEVS	[24]	CDEVS, PDEVS, distributed	RT-DEVS, DSDEVS	
DEVS-Ruby	[9]	CDEVS, PDEVS	DEVStone, flat, decentralized	

tables 2 and 3 continue without: **1**) PyDEVS because newer and future version is PyPDEVS; **2**) AToM^3 which is a multi-paradigm tool where DEVS is supported through PyDEVS; **3**) DEVSimPy which provides a GUI on top of PyDEVS; **4**) SimStudio; and **5**) MS4Me. Many extensions of the formalism have been proposed over time. Table 2 lists supported formalisms for each framework and what particular domains of systems they are best suited for. Table 3 highlights when a repository of usable models is available, if distributed simulations are supported, how modelers can analyze simulations, if a GUI is available and which MDE concepts are supported.

4.2 Performance analysis

For our performance analysis, we compare DEVS-Ruby with aDEVS and PyPDEVS. We retained aDEVS for our comparison because it is written in a compiled, statically typed language (C++) whereas DEVS-Ruby is written with an interpreted, dynamically typed

Table 3 Framework comparison grid based on criteria 5, 6, 7 and 8.

DEVS Frameworks		Criteria				
Name	Ref.	5 Model Sets	6 Network	7 GUI	7 MDE	8 Analysis
aDEVS	[20]	yes	yes	no	no	plot
CD++	[30]	yes	yes	yes	C++ modeler	DEVS view, 2d, 3d, visualization
PowerDEVS	[1]	yes	no	yes	no	quick scope
JAMES II	[38]	yes		yes		plot
VLE	[22]	yes		yes		visualization
PyPDEVS	[24]	no	yes	no		
DEVS-Ruby	[9]	yes	yes	no	DSL	plot, GIS

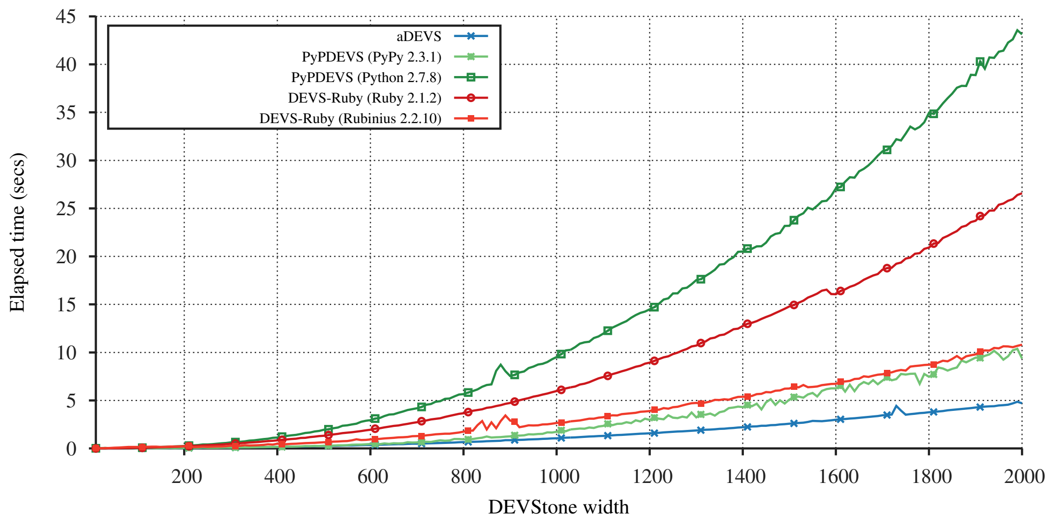

Figure 1 CPU time in seconds of a DEVStone simulation with a varying width and a fixed depth of 3, HI coupling type, δ_{ext} and δ_{int} times set to 0 secs.

language. We used aDEVS performances as an indicator, since [24] found it to be the fastest DEVS framework available. Concerning PyPDEVS, it is written in Python, which is very similar to Ruby. Moreover, DEVS-Ruby is closest to PyPDEVS according to [9].

The test environment is based on an Intel(R) Core(TM) i5-3360M CPU @ 2.80GHz (3MB L2 cache), 16 GB (2 x DDR3 - 1600 MHz) of RAM, a Toshiba MK5061GS hard drive, running on Ubuntu 14.04 (64bit). The software used for the benchmarking are: **1**) DEVS-Ruby 0.6 using the official Ruby VM (version 2.1.2) and an alternative Ruby implementation, called Rubinius (version 2.2.10); **2**) PyPDEVS 2.2.3 using the official Python VM (version 2.7.8) and an alternative Python implementation, called PyPy (version 2.3.1); and **3**) aDEVS 2.8.1 compiled using GCC 4.8.2 with -O3 optimizations flags. To compare performances of selected frameworks (criterium 2), we propose to present the results obtained with a DEVStone [29] benchmark. DEVStone is used to study the performance of DEVS-based simulators. We use it to generate automatically a suite of models with varied structure and behaviour automatically. Using DEVStone, we benefit from a common metric to compare the results obtained using different software. Since it is designed to evaluate the efficiency of DEVS simulation engines, we believe it is the go-to for such a performance analysis.

We measure elapsed wall clock time in seconds for a simulation involving a DEVStone suite of models. Events traverse all models of the generated structure with a fixed *depth* (number of nested coupled models in the hierarchy) of 3, *HI* coupling type (a type of models interconnections defined in [29]) and δ transitions times set to 0 seconds. The varying parameter is the *width* (number of components in each coupled model). All simulators are running the PDEVS formalism. Figure 1 shows the results obtained using the machine specified above. As you can see, the fastest benchmarked simulator is unsurprisingly aDEVS due to its implementation in C++. Concerning DEVS-Ruby and PyPDEVS, DEVS-Ruby is more performant when we are using the official Python and Ruby implementation. But if we switch language implementation to PyPy and Rubinius, both simulators are almost on an equal footing, with PyPDEVS being slightly superior. PyPy and Rubinius are two sophisticated alternative implementations written respectively in Python and in Ruby. They offer great performance thanks to a just-in-time (JIT) native machine code compiler. However, Python code from PyPy is then compiled in C to produce a native interpreter whereas Rubinius VM is written in C++ and nearly everything else is pure Ruby code (lexer, parser, AST to bytecode compiler, standard library). These differences lead Rubinius to poor performances. Another solution consists in using the Topaz Ruby implementation which is built with the same toolchain as PyPy, but it is unfortunately not stable enough yet.

4.3 Discussion

The study of these simulators highlights several points. The PDEVS formalism is more popular in the DEVS community for two reasons: it makes easier to parallelize/distribute simulations and it is more consistent and flexible when it comes to handle simultaneous events. There is no general purpose simulator, each simulator being developed by different institutions working on different aspects and extensions of the formalism, all tending to specialize like CD++ for cellular automata, VLE for agro-ecosystems, James II for agents. Programming languages are important because their syntax and specifications say how much we are ready to make concessions on flexibility and expressiveness, especially to implement models. Studied software either use compiled language (C++), interpreted languages (Python and Ruby) or compiled/interpreted language (Java). In terms of type systems, we have statically typed languages (C++, Java) and dynamically typed languages (Python, Ruby). But this paper emphasizes that the gap in terms of performances between a high performance language like C++ and slower languages like Ruby or Python tend to be reduced thanks to advances in language theory.

5 Conclusion

This paper reports a systematic review of M&S softwares published in journals and conference proceedings. The aim is to propose a survey in the field of simulation software frameworks. Based on an evaluation and selection method, we provide a basis to choose a discrete event simulation framework. Our selection method relies on criteria set related to academic fields. After evaluating several frameworks, we selected three (DEVS-Ruby, aDEVS, PyPDEVS) and we presented a performance benchmark.

Acknowledgements The present work was supported in part by the French Ministry of Research, the Corsican Region and the CNRS. DEVStone implementation for aDEVS and PyPDEVS is based from models originally written by Yentl Van Tendeloo and Prof. Hans Vangheluwe.

References

1 F. Bergero and E. Kofman. PowerDEVS: a tool for hybrid system modeling and real-time. *SIMULATION*, 87(1-2):113–132, January 2011.

2 P.-A. Bisgambiglia, R. Franceschini, F.-J. Chatelon, J.-L. Rossi, and P. A. Bisgambiglia. Discrete event formalism to calculate acceptable safety distance. In *Simulation Conference (WSC), 2013 Winter*, pages 217–228, December 2013.

3 J. S. Bolduc and H. Vangheluwe. A modeling and simulation package for classic hierarchical DEVS. Technical report, 2002.

4 L. Booker, S. Forrest, M. Mitchell, and R. Riolo. *Discrete Event Abstraction: An Emerging Paradigm For Modeling Complex Adaptive Systems by Bernard P. Zeigler Chapter 6 in Perspectives on Adaptation in Natural and Artificial Systems.* Oxford University Press, Oxford, England ; New York, February 2005.

5 L. Capocchi, J.-F. Santucci, B. Poggi, and C. Nicolai. DEVSimPy: A collaborative python software for modeling and simulation of DEVS systems. In *2011 20th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 170–175, June 2011.

6 C. G. Cassandrass and S. Lafortune. *Introduction to Discrete Event Systems.* 2nd ed. 2008. Kluwer Academic Publishers, springer edition, 1999.

7 A. C. Chow, B. P. Zeigler, and D. H. Kim. Abstract simulator for the parallel DEVS formalism. In *, Proceedings of the Fifth Annual Conference on AI, Simulation, and Planning in High Autonomy Systems, 1994. Distributed Interactive Simulation Environments*, pages 157–163, December 1994.

8 J.-B. Filippi, F. Bernardi, and M. Delhom. The JDEVS environmental modeling and simulation environment. *IEMSS, Integrated Assessment and Decision Support, Lugano Suisse*, page 283–288, 2002.

9 R. Franceschini, P.-A. Bisgambiglia, P. Bisgambiglia, and D. R. C. Hill. DEVS-Ruby: a Domain Specific Language for DEVS Modeling and Simulation (WIP). In *DEVS 14: Proceedings of the Symposium on Theory of M&S*, pages 393–398. SCS International, April 2014.

10 S. Gareddu, E. Vittori, J.-F. Santucci, and P.-A. Bisgambiglia. A Meta-Model for DEVS - Designed following Model Driven Engineering Specifications. In *SIMULTECH 2012*, pages 152–157, 2012.

11 N. Giambasi, B. Escude, and S. Ghosh. GDEVS: A generalized discrete event specification for accurate modeling of dynamic systems. In IEEE, editor, *Proceedings of the Fifth International Symposium on Autonomous Decentralized Systems ISADS*, page 464, 2001.

12 J. Himmelspach and A. M. Uhrmacher. Plug'n simulate. In *In Proceedings of the 40th Annual Simulation Symposium (2007)*, pages 137–143, 2007.

13 A. S. Jadhav and R. M. Sonar. Evaluating and selecting software packages: A review. *Information and Software Technology*, 51(3):555–563, March 2009.

14 V. Janoušek and E. Kironskỳ. Exploratory modeling with SmallDEVS. In *Proceedings of the 20th annual European Simulation and Modelling Conference*, page 122–126, 2006.

15 S. Kim, H. S. Sarjoughian, and V. Elamvazhuthi. DEVS-suite: a simulator supporting visual experimentation design and behavior monitoring. In *Proceedings of the 2009 Spring Simulation Multiconference*, page 161. Society for Computer Simulation International, 2009.

16 J. de Lara and H. Vangheluwe. AToM3: A tool for multi-formalism and meta-modelling. In Ralf-Detlef Kutsche and Herbert Weber, editors, *Fundamental Approaches to Software Engineering*, number 2306 in Lecture Notes in Computer Science, pages 174–188. Springer Berlin Heidelberg, January 2002.

17 S. Mittal, J. L. Risco, and B. P. Zeigler. DEVS-based simulation web services for net-centric t&e. In *Proceedings of the 2007 Summer Computer Simulation Conference*, SCSC '07, page 357–366, San Diego, CA, USA, 2007. Society for Computer Simulation International.

18 S. Mittal, J. L. Risco-Martín, and B. P. Zeigler. DEVS/SOA: A cross-platform framework for net-centric modeling and simulation in DEVS unified process. *SIMULATION*, 85(7):419–450, July 2009.

19 J. Nikoukaran, V. Hlupic, and R. J. Paul. Criteria for simulation software evaluation. In *Proceedings of the 30th Conference on Winter Simulation*, WSC '98, page 399–406, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

20 J. Nutaro. ADEVS (a discrete EVent system simulator). *Arizona Center for Integrative Modeling & Simulation (ACIMS), University of Arizona, Tucson. Available at http://www. ece. arizona. edu/nutaro/index.php*, 1999.

21 G. Quesnel, R. Duboz, and E. Ramat. The virtual laboratory environment – an operational framework for multi-modelling, simulation and analysis of complex dynamical systems. *Simulation Modelling Practice and Theory*, 17(4):641–653, April 2009.

22 E. Ramat and P. Preux. Virtual laboratory environment (VLE): a software environment oriented agent and object for modeling and simulation of complex systems. *Simulation Modelling Practice and Theory*, 11(1):45–55, March 2003.

23 H. S. Sarjoughian and B. P. Zeigler. DEVSJAVA: Basis for a DEVS-based collaborative m&s environment. *Simulation Series*, 30:29–36, 1998.

24 Y. Van Tendeloo and H. Vangheluwe. The Modular Architecture of the Python(P)DEVS Simulation Kernel Work In Progress paper. In *DEVS 14: Proceedings of the Symposium on Theory of M&S*, pages 387–392. SCS International, April 2014.

25 L. Touraille, M. K. Traoré, and D. R. C. Hill. A mark-up language for the storage, retrieval, sharing and interoperability of DEVS models. In *Proceedings of the 2009 Spring Simulation Multiconference*, SpringSim '09, page 163:1–163:6, San Diego, CA, USA, 2009. Society for Computer Simulation International.

26 L. Touraille, M. K. Traoré, and D. R. C. Hill. SimStudio : une infrastructure pour la modélisation, la simulation et l'analyse de systèmes dynamiques complexes. Technical Report RR-10-13, INRIA, May 2010.

27 M. K. Traoré. SimStudio: A next generation modeling and simulation framework. In *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, Simutools '08, page 67:1–67:6, ICST, Brussels, Belgium, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

28 H. Vangheluwe. The discrete EVent system specification DEVS formalism. Technical report, Modeling and Simulation (COMP522A), 2001. Published: Lecture Notes http://moncs.cs.mcgill.ca/.

29 G. Wainer, E. Glinsky, and M. Gutierrez-Alcaraz. Studying performance of DEVS modeling and simulation environments using the DEVStone benchmark. *SIMULATION*, 87(7):555–580, July 2011.

30 G. A. Wainer. CD++: a toolkit to define discrete-event models. *Software, Practice and Experience. Wiley*, 32(3):1261–1306, November 2002.

31 G. A. Wainer and N. Giambiasi. Application of the cell-DEVS paradigm for cell spaces modelling and simulation. *SIMULATION*, 76(1):22–39, January 2001.

32 G. Zacharewicz, M. El-Amine Hamri, C. Frydman, and N. Giambiasi. A generalized discrete event system (g-DEVS) flattened simulation structure: Application to high-level architecture (HLA) compliant simulation of workflow. *SIMULATION*, 86(3):181–197, March 2010.

33 B. P. Zeigler. *Guide to Modeling and Simulation of Systems of Systems - User's Reference*. Springer Briefs in Computer Science. Springer, 2013.

34 B. P. Zeigler, G. Ball, H. Cho, J. S. Lee, and H. S. Sarjoughian. The DEVS/HLA distributed simulation environment and its support for predictive filtering. Technical report, 1998.

35 B. P. Zeigler, S. B. Hall, and H. S. Sarjoughian. Exploiting HLA and DEVS to promote interoperability and reuse in lockheed's corporate environment. *SIMULATION*, 73(5):288–295, November 1999.

36 B. P. Zeigler, H. Praehofer, and T. G. Kim. *Theory of Modeling and Simulation, Second Edition*. 2000.

37 B. P Zeigler and H. S. Sarjoughian. DEVS integrated development environments. In *Guide to Modeling and Simulation of Systems of Systems*, page 11–26. Springer, 2013.

38 S. Zinn, J. Himmelspach, A. M. Uhrmacher, and J. Gampe. Building Mic-Core, a specialized M&S software to simulate multi-state demographic micro models, based on JAMES II, a general M&S framework. *J. Artificial Societies and Social Simulation*, 16(3), 2013.