DEVSimPy: A Collaborative Python Software for Modeling and Simulation of DEVS Systems

Laurent Capocchi, Jean-François Santucci, Bastien Poggi, Céline Nicolai

DEVSimPy:A Collaborative Python Software for Modeling and Simulation of DEVS Systems

L. Capocchi, J.F. Santucci, B. Poggi, C. Nicolai
University of Corsica, UMR CNRS 6134 SPE
Corte, France
{capocchi,santucci,bpoggi,cnicolai}@univ-corse.fr

Abstract—**The Modeling and Simulation (M&S) of complex systems leans on the collaboration between different actors coming from specific domains. These actors have to communicate through an efficient software in order to improve the M&S process. We therefore propose in this article a collaborative M&S software framework called DEVSimPY. We point out the use of DEVSimPy through a concrete case study: hydraulic network management.**

Keywords-Modeling; Simulation; Collaborative softwares; Discrete event systems; Hydraulic systems; Software libraries

I. INTRODUCTION

The design of models for the simulation of complex systems is a task requiring a lot of time to be achieved. One way to speed up this task is to develop methodologies for: (i) deriving and using reusable design components; (ii) developing collaborative modeling and simulation. Recently a set of research work has been oriented towards this direction. We can highlight that traditional modeling and simulation approaches are mono-disciplinary in style, in that the simulation software development concerns only one application domain. However simulating complex systems, in which the viability of the simulation software depends on the close coupling between different disciplines, therefore calls for a more multidisciplinary approach. In the same time, computer simulations have been becoming increasingly complex and require efficient system simulation framework. Furthermore increasing size and geographical separation of design data and teams has created a need for network-based design environments [1,2,3]. Usually modeling and simulation tools are associated with libraries of reusable modeling components that will make the description of the models and also their validation much easier [4,5,6]. Storing models in a common generic library has several benefits. First, the genericity of this storage service can be offered to various modeling and simulation environments. Second, a common library allows environments to share information so they can interact each other, and third, several users can share modeling components. This last point is the most important since it allows a modelling and simulation team enabling an efficient collaborative work. Modeling and Simulation applications of today and tomorrow will be increasingly based on three fundamental technologies: object orientation, Collaborative Design and Internet (especially the World Wide Web):

- object orientation allows applications to be viewed in terms of natural objects,
- collaborative testing help,
- and the Web allows to access resources located all around the world. A Web-based access is a very interesting perspective for a generic modeling and simulation components library, since it allows a user-friendly remote access using a simple web browser.

Three distinct levels of communication with a Computer Modeling and Simulation framework have to be considered. At the higher level of communication, a domain specialist interacts with the Computer Modeling and Simulation framework in order to define an environment contributing to the solution of a certain class of problems related to a given domain. We call this environment an application system and the person who defines it an Application Builder. His role is to define appropriate models for the class of applications, specify appropriate processing tools and establish problem resolution strategies adapt to his class of applications. At the lower, an application system is activated by a user (called the Final User) in order to solve a particular problem. The role of the Final User is to supply the problem description and other relevant information to the application system.
The remaining level correspond to the Framework Builder, who: (i) is familiar with the internal software structures of the Computer Aided Design framework and (ii) is in charge of the creation and management of the different libraries involved in application domains.

In this paper we propose to develop a Collaborative Modeling and Simulation Framework based on DEVS (Discrete Event Specification) simulations. The DEVS formalism introduced by Zeigler [7] provides a means of specifying the behavior of a given system. It provides a formal representation of discrete event systems capable of mathematical manipulation just as differential equations serve this role. One of the main interests in DEVS formalism is the fact that it allows an explicit separation between the modeling and simulation part. This means that we can define the model representing the behavior of a given system without having to consider the simulation phase.
The definition of an efficient collaborative modeling and simulation framework raises a set of problems. The first type of problems concerns the manipulation of the basic elements (atomic models, coupled models and test benches). It is a very

important part of simulation models Design. The elements are normally defined in terms of concepts which are specific to the class of applications for which the model is used. These definitions are stored in a library. When an element occurs in a model description, its definition is copied from the library and placed in the context of the model. Thus an element once defined and placed in the library, can be used as often as required without having to be redefined. The second type of problems to solve is to propose the management of software design errors using a web based access. The idea is that the final user could have the possibility to automatically refer to the Application Builder when a model design error happens and in the same way that the Application Builder could refer the Framework Builder when a software framework error happens. The last type of problems to be solved concerns the ways to share both atomic models, coupled models and testing developments.

We describe in this article an object-oriented framework called DEVSimPy allowing to propose solution to the three types of problem previously mentioned. Furthermore this environment provides a user-friendly interface for creating DEVS models. The simulation is performed by running the PyDEVS simulator[8]. The originality of our approach lies in the facts that it is based on a strong notion of genericity of use. We have already validated the use of the proposed approach in the framework of the simulation of a complex system: the software simulation of the management of a hydraulic network. The design of the software leans on the collaboration of several specialists in charge of different part of the common library of models to be defined : (i) a specialist of the development of mathematical modelling in charge of the simulation of the behaviour of dams, (ii) a specialist of spatial data modelling in charge of the simulation of data related to geographic information systems, (iii) a specialist of discrete event modelling in charge of the simulation of the hydraulic Engineer strategies and (iv) a specialist of learning based models in charge of the simulation of predictive systems as rainfall or water consumption. These four types of specialists will be able to collaborate within the proposed software framework as representing each of them an Application Builder. We will point out how the DEVSimPY framework has been used in order to define a software simulation of the hydraulic network system.

The paper is organized as follows. Section 2 presents the basics of our work: the DEVSimPy framework. Section 3 first introduces the complex application case: the simulation of an hydraulic network management. We then points out how the DEVSimPy framework has been used in order to perform a simulation model of such complex system.
Finally, section 4 concludes the paper and provides some perspectives of work.

II. DEVSIMPY M&S SOFTWARE

A. *Initial idea*

In this section we present a software framework dedicated to DEVS modeling and simulation. This framework is being developed at University of Corsica using the Python Language. The SPE (Sciences Pour l'Environnement) laboratory of the University of Corsica is specialized in the field of environmental systems modeling based on DEVS formalism. DEVS is a mathematical formalism allowing modeling and simulation of discrete event systems in hierarchical and modular way [9]. In the DEVS formalism, one must specify: 1) basic models from which larger ones are built (they are called atomic models), and 2) how these models are connected together in hierarchical fashion (to obtain coupled models). The SPE team has published several works on extensions of the DEVS formalism responding to problems from different application areas as:

- the simulation of forest fires,
- the behavioral test of digital circuits,
- the behavioral simulation of hydraulic network,
- the simulation of sensor networks,
- the simulation of myths and legends,
- …

When a member of SPE team has to define a DEVS model of a system, he always uses a generic and reusable approach. The choice of the best programming language used in order to implement the model depends on factors in relation with the nature of the system. In recent years, the PyDEVS library has been increasingly used in the SPE laboratory due to its simplicity of use and also because it is coded in Python[10]. Python is an object-oriented programming language known to be clear and readable, fast to learn, reusable and portable on multiple platforms. PyDEVS provides an API allowing the user to model and simulate DEVS models in Python code. This API is used in the excellent multi-modeling GUI software named ATOM3 [11] which allows to use several formalisms without focusing on DEVS.

The initial idea of DEVSimPy (Python Simulator for DEVS models) project is to provide a GUI based on PyDEVS API in order to facilitate both the coupling and the reusability of PyDEVS models. It is a user-friendly interface for collaborative modeling and simulation of DEVS systems implemented in Python. With DEVSimPy the modeler can:

- model a DEVS system and save it into a library,
- edit the DEVS model during the simulation,
- import existing library of models,
- automatically simulate/analyze the system,
- …

DEVSimPy is an open source project under GPL V3 license and the SPE research laboratory team supports its development. It uses the wxPython graphic library, the Python Language and the PyDEVS API (Fig. 1). It can be downloaded from http://code.google.com/p/devsimpy/.

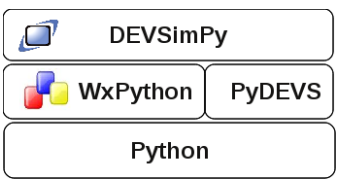

Figure 1. DEVSimPy Layer View

B. *DEVSimPY features*

DEVSimPy has been built to facilitate collaboration of researchers during the DEVS modeling. Indeed, DEVSimPy final users (Fig. 2) can develop and maintain libraries of generic components. The libraries can be stored on the hard disk or network to improve sharing. In addition, the components belonging to dynamic libraries are compressed files that contain the behavior and graphic of the models. This approach keeps the separation between the behavior and the view of the models. So DEVSimPy framework builder improves the sharing of component libraries making an easy manipulation. The M&S kernel has been build by the DEVSimPy application builder in order to make possible the simulation of models based on the components.

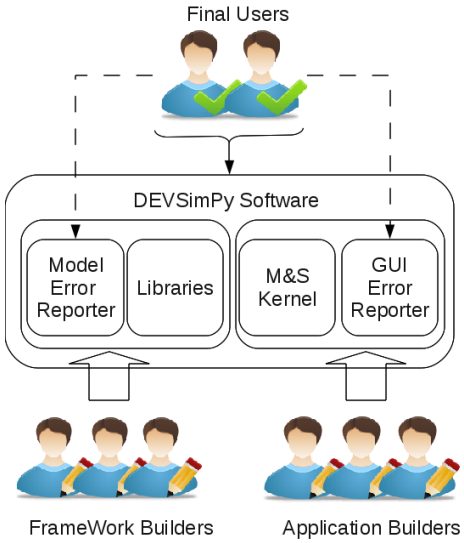

Figure 2. DEVSimPy actors and collaborations

When a final user debugs its models, an error can occurs. The framework and application builders have developed two error managers (Fig. 2). These modules provide to the final users an easy way to correct their models or to send emails to the authors of the models. If the error occurs in the kernel, the GUI error reporter sends emails to DEVSimPy developers.

1) *Sharing Component Dynamic Libraries*

In DEVSimPy, libraries are directories that contain components represented by files. Users can create libraries which can be stored on a local hard disk or on an external http server. Thus, there is two ways to share components: read-only when the shared library is stored on an http server and read/write when the shared library is stored on the hard disk.

This mode of sharing allows to create two collaborative approaches depending on the nature of DEVSimPy users: developer for the read/write access or single user for read-only access. DEVSimPy has a library manager with a specific interface that offers several features (import, activation, deactivation,...). All imports of new libraries in DEVSimPy are done through this interface. The user can also view the libraries already imported with their information such as size, location and file name.

2) *Handling components and diagrams*

A library is composed by files used in order to instanciate components. These components must be portable and easily usable in DEVSimPy. That is why they are constructed from compressed files that contain: a behavioral/structural file for atomic/coupled DEVS model and a graphic file. In this way, there is a separation between behavior (structure) and representation of models.

DEVSimPy has two file extensions depending on the nature of the described components:

- .amd for atomic models: This compressed file consists of a behavior (devs transition functions) and a graphic file (color, shape ,...).
- .cmd for coupled models: This file consists of a file defining a structure (list of DEVS models, coupling between DEVS models,...) and a graphic file. The behavioral or structural files have a file extension ".py" and the graphic files a ".dat".

DEVSimPy was built to facilitate files handling and sharing implemented in PyDEVS. Therefore, a component can be an uncompressed python file. In this case, its representation is automatic in DEVSimPy. However, the user can export it in .amd or .cmd file in order to customize and save its graphic.

To facilitate the use of components, DEVSimPy allows automatic documentation templates. A management system can automatically generate model documentation from python source code. DEVSimPy also offers the debugging of models before the simulation with the possibility of modifying the source code using an embedded code editor.

An error manager is implemented in DEVSimPy in order to inform (mainly by email) developers of an error in the software kernel. This module manages also errors coming from the simulation of models by sending an email to the authors. Its also possible to edit the code source in order to correct the errors.

Another way to have a good collaboration between all actors of the modeling is to use a plugins approach. In fact, each specialist of the domain can in an easy way make a code extension (usually called plugin) that can help the final user in the manipulation of the models. For example, developers can write plugins that show the simulation results in graphs. DEVSimPy is based on a powerful plugin architecture managed by a specific interface allowing activation/desactivation and setting of all installed plugins.

When the user manipulates several components it builds a diagram. These sets of interconnected components can be stored in a compressed file with the .dsp (for DESVSimPy) file extension. Collaboration for a modeling project can be effected through the exchange of such files.

In summary, using DEVSimPy the developer can share components in an efficient way. It can also ask a user to test the simulation models sending both the http link of the

component library (or compressed libraries) used by the models and the .dsp file representing the full model.

3) Facilitating the simulation process

Like with DEVS, DEVSimPy simulation algorithms are is automatically generated. Once the model has been built, simulation can be start by clicking on a simple button. A dialog appears giving access to several options like the verbose simulation, the choice of simulation algorithm, the choice of profiling simulation and the final simulation time.

The simulation process is build with a threading approach. Each simulation is generated from a model and several simulations can be performed in parallel. This parallel simulation engine embedded in DEVSimPy is an advantage because user can simulate several behaviors from a single model. Moreover, several simulations can be performed in a concurrent–way from several different models. In this way, sharing data is possible between each model.

DEVSimPy allows to go further by using the dynamic simulation property allowing the change of the model behavior during the simulation. Whichever the simulation time, the user can suspend the simulation and edit the code of models in order to modify their behaviors. An another way to do it, is to replace the python file describing the model behavior. In both case, the simulation process can then restart with the new behaviors without any alterations. This aspect is one of the most important aspects because the user can interact with the behavior of models without waiting the end of the simulation.

Another simulation tool proposed by DEVSimPy is the profiling. The modeler can perform the simulation with a profiling option for measuring where models consume resources, including CPU time and memory. Each diagram can be profiled and DEVSimPy save their profiling results in separate files. Profiling can be painfully slow and the user should use it with precautions.

Two types of errors manager are proposed in DEVSimPy. The first one is dedicated to the errors coming from DEVsimPy kernel code. The second manages the errors coming from the code of the models. The latter is interesting from the standpoint of user because the user is informed when a syntax error occurs in the code. The error checking is possible in many case: before the simulation, during the coding of models and during the importation of libraries. When an error is detected, DEVSimPy offers a specific dialog asking to the user if he wants to correct this error.

Finaly, DEVSimpy is composed by several models that are dedicated to the view of events and states (QuickScope, MessageViewer, EventViewer). These models are stored in library and can be used by the user irrespective of the study domain. Their main goal is to help the developer to validate models or to analyze the intermediate signals and events under the modeling process.

III. CASE STUDY: HYDRAULIC NETYWORK M &S

A. Description

The members of the SPE team from University of Corsica are collaborating on a modeling and simulation tool. Our goal is to improve the hydraulic network located in the south of the Corsica Island. Our work comes within the scope of a European cross-border project. The main difficulty is to make an easy collaboration between different specialists in different fields. This scientific sphere can be: computing, mathematics, physics, hydrology and management. The application builders need to communicate and to study the hydraulic network to get realistic behavioral specifications (inputs of the system). With these specifications they can build a determinist model. This one can be simulated to propose an optimization of hydraulic network management (output of the system).

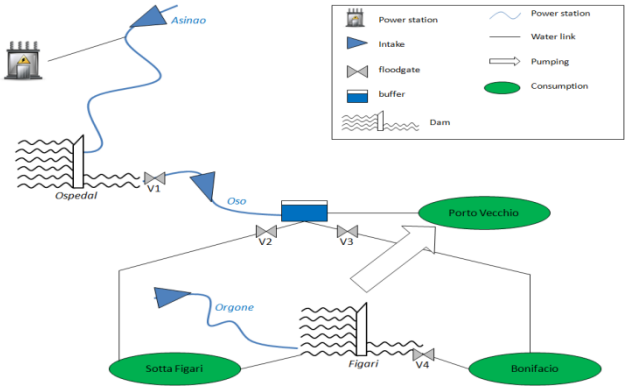

Figure 3. Scheme of the Hydraulic network under study

On Fig. 3 we can see a global system view to model. Outcome of analysis is this plan which represents the general network behavior. Some component like dams, intakes, water link are clearly identifiable and are essential to find optimization at the end of simulation. We have defined three simulation scenarios:

- Maximize water-generated electricity: As you can see on Fig. 3 the network has a power station. This electrical production represents a loss of water but it saves a considerable amount of money.
- Minimize pumping between dams: The water demand increases faster during the tourist season. This brings about water pumping to carry out supplying customers with water.
- Improve the floodgates management (minimal pressure, stock distribution): A minimal pressure needs to be supported over the hydraulic network. It depends on the water stock and floodgate turning on.

Our goal is to allow simulation of complex scenarios and help the decision-maker to take the best choice according to collected data (weather, consumption, stocks).

The system modeling is complex and includes some type of sub-systems:

- Data system based on statistic tools like neuronal networks or data mining.

- Behavioral system based on literal or oral description from the expert.
- Mathematical system based on equation and mathematical formula like differential equation or final state automata.

All these sub-systems are complementary and we would like to use them to build a global system that is coherent with the real hydraulic network functioning as we describe. The collaboration on this project is the key of success and the software DEVSimpy is collaboration oriented.

B. Collaborative M&S Process

The development of a discrete-event modeling and simulation software requires the collaboration of different actors at the main following three steps [12]: (i) Specification task, (ii) Model Design and (iii) simulation and validation.
In this sub-section we point out how the DEVSimPy software framework has been used in order to deal with the case study presented above.

1) Specification Task

The specification task concerns the definition of modeling objectives and requirements analysis. We have performed this first step without any help of the DEVSimPy framework since this task has to elaborate the modeling objectives and requirements without any link with the DEVS formalism. The result of this first step has been obtained after a set of meetings with the Hydrologic Engineers. The result consists in a list of literal descriptions of basic behavior of the different parts composing the overall system under study and presented in section III.A.

2) Model Design Task

This task concerns the design of the set of DEVS models which will be integrated under an overall coupled model. We have to define which models have to be created and how they have to be interconnected. This resulting interconnection is then simulated. The implementation has been performed using the DEVSimpy framework. We have used the framework in order to facilitate the design of the basic models and their interconnection when dealing with the DEVS simulation of the Hydraulic Network described in sub-section III.A. The implementation has been facilitated by the following three features of the DEVSimPy framework: (i) automatic documentation of the code, (ii) the development of libraries and (iii) the automatic generation of code.

The self-documentation of the code is performed during code creation of the models. The application builders have the possibility to add comments by respecting Epytext markup language formalism [13]. These comments can be parsed by the Epydoc engine and are used to generate a fulfilled centralized and structured documentation.

The development of libraries allows the Application Builder to offer a set out-context of atomic or coupled models that can easily be inserted in a global simulation models. In the case of the considered application (simulation of an hydraulic network), four different specialists are required in order to define the basic elements : (i) a specialist of the development of mathematical modelling describing the behaviour of dams, (ii) a specialist of spatial data modelling related to geographic information systems, (iii) a specialist of discrete event modelling concerning the hydraulic Engineer strategies and (iv) a specialist of learning based models associated with predictive systems as rainfall or water consumption. Each of them has defined a set of basic atomic models (saved as .amd files) and coupled models (saved as .cmd) that are going to be interconnected in an overall coupled model: (i) the first specialist has defined 6 atomic models (stored as .amd in the library) - pumping atomic model, electrical Power Station atomic model, dam behaviour atomic model, water releases atomic model, pipeline atomic model, valve behaviour atomic model; (ii) the second one has defined three atomic models - point atomic model, line atomic model and polygon atomic model; the third one has defined three atomic models corresponding to the different scenarios involved by the management of an hydraulic network – turbine management atomic model, pumping management atomic model and valve management atomic model; and finally the last specialist has defined two atomic models – rainfall prevision atomic model, water consumption atomic model.

In order to facilitate the use of these atomic models we have defined a set of coupled models stored in the library under the .cmd format. These coupled models have been obtained by interconnection of some of the previously mentioned atomic models: dam location coupled model, and dam behaviour coupled model.

A third kind of element can be stored in the library: the .dsp models. They correspond to validation scenarios that the different specialists can defined in order to test their respective models. Furthermore the final user can also defined a .dsp model corresponding to the actual water network he has to simulate in order to obtained information concerning its management. The part 3 of section III.B will detail this simulation and validation process
The last feature offered by DEVSimPy in order to help the Application Builder to design the models to be stored in the library is the automatic code generation tool. DEVSimPy offers the possibility to automatically generate a set of parts of code corresponding to repetitive coding. For example the code corresponding to the checking of arrival of an event is automatically generated on demand of the Application Builder.

3) Simulation and Validation Task

In this part we show the simulation results through the QuickScope atomic model in three points: (1) Asinao water flow output, Electrical Power station output and the dam output.

The use of the QuickScope atomic model allows to validate the signal form of the water flow in the system. The simulation time step is the week and the context of the simulation is as follows (Fig. 4): the water flow of catchment basin 'Asinao' decreases between the week 1 and 30 and increases between the week 30 and 52. The electrical Power atomic model is operational if the level of water flow coming from Asiano is bottom at 50000 m3/week. In the other case the dam retains the water.

The three following figures (Fig. 4, Fig. 5, Fig. 6) show the operations of the system and validate its behavior. Fig. 4 represents the variation of water coming from the catchment basin 'Asinao'. Fig. 5 represents the Hydropower generate by the electrical power station depending on the data of the Asinao water flow (Fig. 4). Fig. 6 represents the filling up of the dam. When the dam is stable, the electrical micro-central is indeed operational.

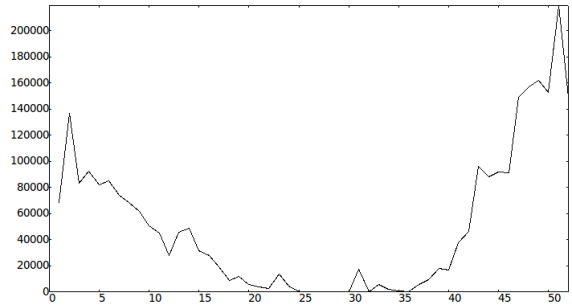

Figure 4. Asinao water flow [m3/week]

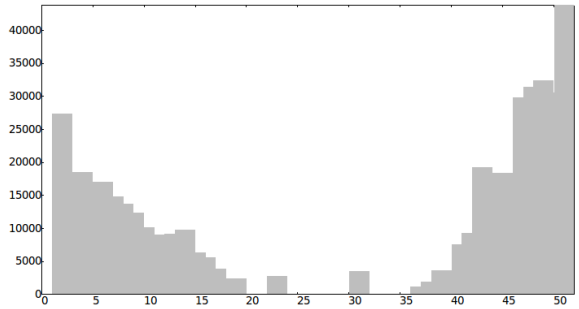

Figure 5. Electrical Power Station [Watt/week]

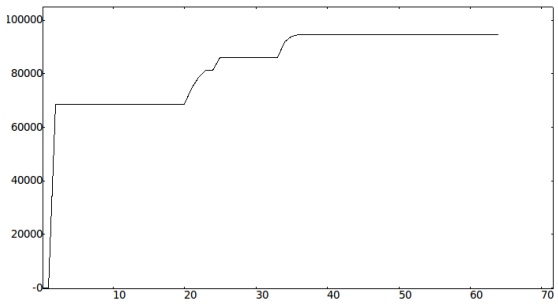

Figure 6: Dam filling up [m3/week]

IV. CONCLUSIONS AND FUTURE WORK

We introduced in this paper our approach for the definition of a collaborative modeling and simulation framework: DEVSimPY. This tool enables the reusability and sharing of modeling components, an efficient management of different kinds of errors and a friendly test process of the designed models. The proposed framework allows the model designer to perform its model design very quickly, once models have been introduced in a library. We provided also this user with a set of APIs allowing him to process his models through some web-based APIs and graphical interfaces accessible using the DEVSimPy framework.

Our approach has been validated in the framework of a concrete and complex domain: the simulation of an hydraulic network management.

We have three main perspectives of work. The next step will be to develop the possibility to associate several kinds of simulation associated with the library stored models. The second perspective is to define a meta-language in order to help the models design. Finally, we want to study how to manage and maintain with the most efficiency a distribution of simulation models over a network and over multiple hosts.

REFERENCES

[1] H. Praehofer, J. Sametinger, and A. Stritzinger, "Building Reusable Simulation Components," presented at WebSIM2000, Web-Based Modelling and Simulation, San Diego, CA, USA, 2000.

[2] Filho, W. A., Hirata, C. M. and Yano, E. T. (2004). GroupSim: A collaborative environment for discrete event simulation software development for the World Wide Web, SIMULATION, Vol. 80, No. 6, 257-272.

[3] D.J. van der Zee, "Developing participative simulation models: framing decomposition principles for joint understanding", Journal of Simulation, vol. 1, pp. 187--202, 2007.

[4] O. Balci, A. I. Bertelrud, C. M. Esterbrook, and R. E. Nance, "Developing a Library of Reusable Model Components by Using the Visual Simulation Environment," presented at the 1997 Summer Simulation Conference, 1997.

[5] A. P. J. Breunese, J. L. Top, J. F. Broenink, and J. M. Akkermans, "Libraries of Reusable Models: Theory and Application," *Simulation*, vol. 71, 1998.

[6] B. Meyer, "Reusable Software - The Base Object-Oriented Component Libraries", *Prentice Hall Professional Technical Reference*, 1997.

[7] B.P. Zeigler, "*Theory of Modeling and Simulation"*, New York: Wiley, 1996.

[8] J.S. Bolduc, H. Vangheluwe, "*The modeling and simulation package PythonDEVS for classical hierarchical DEVS"*, Technical report MDSL-TR-2001-01, McGill University, Montréal, Canada, 2001.

[9] B. P. Zeigler, T. G. Kim, and H. Praehofer, "Theory of Modeling and Simulation.", Orlando, FL, USA: Academic Press, Inc., 2000.

[10] M. F Sanner, "Python: a programming language for software integration and development," *J. Mol. Graphics Mod* 17, pp. 57–61 , 1999.

[11] J. de Lara and H. Vangheluwe, "AToM3: A Tool for Multi-formalism and Meta-modelling.", In *Proceedings of the 5th International Conference on Fundamental Approaches to Software Engineering* (FASE '02), Ralf-Detlef Kutsche and Herbert Weber (Eds.). Springer-Verlag, London, UK, UK, pp. 174--188, 2001.

[12] Chang Ho Sung, Il-Chul Moon, Tag Gon Kim, "Collaborative Work in Domain-Specific Discrete Event Simulation Software Development: Fleet Anti-air Defense Simulation Software," wetice, pp.160--165, 2010 19th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, 2010.

[13] E. Loper, "Epydoc: API documentation extraction in Python," *URL: http://epydoc. sourceforge. net/pycon-epydoc. ps. Accessed* 13 (2008).