RESEARCH-ARTICLE

Benefits and challenges in developing warship simulator based on DEVS formalism

CHANG BEOM CHOI, Korea Advanced Institute of Science and Technology, Daejeon, South Korea

JANGWON BAE, Korea Advanced Institute of Science and Technology, Daejeon, South Korea

TAG GON KIM, Korea Advanced Institute of Science and Technology, Daejeon, South Korea

JAEICK KIM, Agency for Defense Development, Daejeon, South Korea

WOONG GIE HAN, Agency for Defense Development, Daejeon, South Korea

CHEOL HO KIM, Agency for Defense Development, Daejeon, South Korea

Open Access Support provided by:

Agency for Defense Development

Korea Advanced Institute of Science and Technology

Published: 11 April 2010

SpringSim '10: 2010 Spring Simulation Conference
April 11 - 15, 2010
Florida, Orlando

https://doi.org/10.1145/1878537.1878607
ISBN: 9781450300698

Benefits and Challenges in Developing Warship Simulator Based on DEVS Formalism

Chang Beom Choi, Jangwon Bae, Tag Gon Kim
KAIST, 373-1 Guseong-dong, Yuseong-gu, Daejeon 305-701, Republic of Korea
{cbchoi, jwbae}@smslab.kaist.ac.kr, tkim@ee.kaist.ac.kr

Jaeick Kim, Woong Gie Han, Cheol Ho Kim
the Agency for Defense Development, Jinhae, Republic of Korea
{jaeick, hahaha, aceman}@add.re.kr

Keywords: Discrete Event Simulator Verification, Requirement Specification, DEVS Formalism

Abstract

Modeling and simulation(M&S) engineering is one of the most challenging areas that have to deal with problems from multiple domains. Hence, in the M&S field, the various domain experts and the M&S experts often work together to build a simulator. Yet, in some domains like military, cooperation has been limited because of the security policies in domains. Therefore, the domain experts in such fields are required to have M&S knowledge on the top of their domain knowledge to build simulation models by themselves.

This paper describes our experience of developing a simple warship simulator and assisting such domain experts to obtain the M&S knowledge using Warship Simulator Project. From the experience of the project, we found that the DEVS formalism is easy to learn, and it is suitable for developing a simulator easily with implementation of DEVS formalism.

1. INTRODUCTION

Computer simulation,which is called a computer model or a computational model, is a computer program or a network of computers that attempts to simulate an abstract model of a particular system. Computer simulation has become a useful method of mathematical modeling of many natural systems in physics (computational physics), chemistry, biology, human systems in economics, psychology, and social science and in the process of engineering new technology, to gain insight into the operation of those systems, or to observe their behavior[1].

In order to build a simulator, the developer of the simulator should consider the objectives, e.g. abstraction of the simulation models, of the simulation. for example, In the realtime simulations, simulation models of the simulator should operate within the deadline, therefore, abstract simulation models are required in the realtime simulations. However, simulators for analyzing the target systems should simulate target system accurately without computational deadline.

Once modeling objectives are decided, modeling and simulation(M&S) experts can describe only the abstract level of the target system. to describe more lower level of target system, M&S experts should have deep knowledge of the target system. However, this is not always the case. On the other hand, domain experts - e.g. researchers in national defense research institute - have detailed knowledge of the target system. However, they do not have M &S knowledge to develop abstract models of target system which meets the simulation objectives. Therefore, modeling requires cooperative teamwork between domain experts and M&S experts in a whole modeling process. There have been some efforts for communicating and co-operation between domain experts and M&S experts using Unified Modeling Language (UML)[2]. In this approach, domain experts analyze the system specification using UML,develop several diagrams of UML, and then M&S experts transform the diagrams to simulation models with additional information which is necessary for discrete event simulation. In the case of the military field, most of the information of the system, such as system behavior, protocol, and doctrines, is not opend to the M&S experts. Therefore,they can not develop accurate models which meets the modeling objectives. On the other hand, domain experts are required to have M&S knowledge on the top of their domain knowledge to build simulation models by themselves. The Agency for Defense Development(ADD) launched Warship Simulator Project[1] with System Modeling and Simulation Laboratory(SMSLAB) in Korea Advanced Institute of Science and Technology(KAIST). To overcome such problems, we applied Co-simulation methodology in our new project[3]. The objectives of this project are two fold; one is building warship simulator to measure the effectiveness of the warship and its equipments and the other is assisting the researchers in ADD to build their own simulation model. In order to achieve these goals, M&S experts of SMSLAB and the domain experts in ADD cooperate together based on the Discrete Event Systems Specification (DEVS) formalism.

[1]This research is funded by Agency for Defense Development. The information of the project is classified at the CONFIDENTIAL level. Therefore, we use the name Warship Simulator Project instead.

2. DEVELOPMENT METHODOLOGY ADOPTED IN WARSHIP SIMULATOR PROJECT

As we mentioned earlier, the objectives of Warship Simulator Project are to build a simulator, and assist the researcher to obtain M&S knowledge so that they can build their own simulation model. In order to build a warship simulator, we adopted the Co-modeling Methodology. In this section, we briefly review the co-modeling methodology proposed in [3].

2.1. Modeling Process using UML and DEVS

The co-modeling methodology is analogous to HW/SW co-design in VLSI systems design. HW/SW co-design means the meeting of system-level objectives by exploiting the trade-offs between hardware and software in a system through their concurrent design. To do so, system specification is partitioned in hardware and software parts for concurrent job. After that, each part is implemented and then integrated for co-simulation[4][5]. Co-modeling methodology is analogous to HW/SW co-design and the process is shown by Figure1.

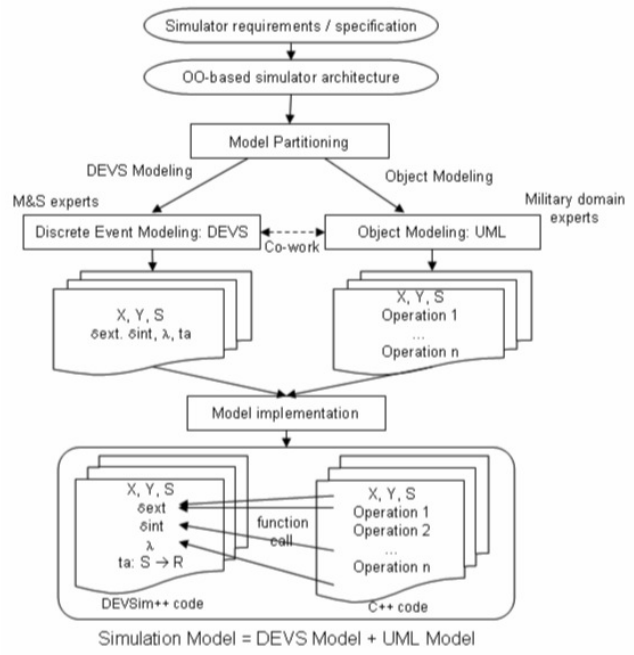

Figure 1. Co-modeling methodology using UML and DEVS[3]

At first, we design the simulator architecture from requirements and specification for a system to be simulated. Mostly, M&S experts do not need detailed knowledge on an object in atomic DEVS modeling, they define three sets and four functions according to DEVS formalism. On the other hand, if atomic DEVS modeling needs detail knowledge on the object, domain experts define such knowledge in forms of operations on the object using UML. Then, M&S experts employ the operations as services. Specifications of DEVS and UML models are implemented by DEVSim++[11] and C++, respectively.

2.2. Advantages of applying Co-modeling methodology

Within the proposed methodology, M&S experts and domain experts design simulation models in a closely-coupled manner. Co-modeling methodology clearly gives M&S experts how they can change behaviors of the model. In the co-modeling methodology, M&S experts and domain experts partition the target system into two parts; discrete event modeling and object modeling. Discrete event modeling deals with abstracted level of behaviors, mostly in state-transition level and Object modeling treats domain-specific calculation and decision making.which indicates object modeling. In order to transit between states, decision making or calculation that involves domain-specific knowledge are necessary in most cases. With the structure of separating discrete modeling and object modeling, M&S experts do not need specific knowledge of objects which only domain experts understand. Domain experts may concentrate the operations of an object in detail. In this sense, DEVS modeling is viewed as software and UML modeling viewed as hardware in the HW/SW co-design methodology. Thus, the main advantage of the co-modeling methodology is a concurrent process in models development.

3. UML AND DEVS FORMALISM: BRIEF REVIEW

In this section, we briefly introduces the UML and DEVS formalism used in Warship Simulator Project.

3.1. UML Modeling

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, and documenting the artifacts of an object-oriented system under development[6][7][8]. It simplifies the complex process of software design, making a blueprint for construction. This sub-section describes only three diagrams in UML, which are used for modeling of an object of the warship simulator. They are use case diagram, class diagram, and sequence diagram.

3.1.1. Use Case Diagram

A use case diagram is a behavior diagram that defines a set of use cases and actors and relationships between them. As a behavioral classifier the diagram defines a sequence of actions, performed by one or more actors and a system, which

results in an observable value to one or more actors. For system developers, this is a technique for gathering system requirements from a user¡¯s point of view.

3.1.2. Class Diagram

A class diagram is a structure diagram that shows a set of classes, interfaces, and/or collaborations and the relationships among these elements. A class includes name, attributes and operations. This diagram is a central modeling technique that runs through nearly all object-oriented methods and represents the static part of a system.

3.1.3. Sequence Diagram

A sequence diagram is an interaction diagram that focuses on the time-ordering of a message between objects. A sequence diagram depicts a sequence of actions that occur in a system which is a very useful tool to easily represent the dynamic behavior of the system. This diagram includes objects and messages in two-dimensional form in nature. On horizontal axis, it shows the life of objects that it represents, while on the vertical axis, it shows the sequence of the creation or invocation of these objects.

3.2. DEVS Formalism

The DEVS formalism is a set-theoretic formalism which specifies discrete event models in a hierarchical and modular form[9]. With this formalism, one can perform modeling more easily by decomposing a large system into smaller component models with coupling specification between them. There are two kinds of models: atomic model and coupled model.

An atomic model is the basic model and has specifications for the dynamics of the model. Formally, a 7-tuple specifies an atomic model M as follows.

$$AM = < X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta >$$

where

X : a set of input events,
Y : a set of output events,
S : a set of sequential state,
δ_{ext}: $Q \times X \rightarrow S$, an external transition function
where Q is the total state set of
$Q = \{(s,e) | s \in S \, and \, 0 \leq e \leq ta(s)\}$,
δ_{int}: $S \rightarrow S$, an internal transition function,
λ: $S \rightarrow Y$, an output function,
ta: $S \rightarrow \mathcal{R}^{+}_{0,\infty}$, a time advance function,
where the $\mathcal{R}^{+}_{0,\infty}$ is the non-negative real numbers with ∞ adjoined.

A coupled model provides the method of assembly of several atomic and/or coupled models to build complex systems hierarchically. Formally, a coupled model is defined as follows.

$$CM = < X, Y, M, EIC, EOC, IC, SELECT >$$

where

X : a set of input events;
Y : a set of output events;
M : a set of all component models;
$EIC \subseteq CM.X \times \cup M.X$, external input coupling;
$EOC \subseteq \cup M.Y \times CM.Y$, external output coupling;
$IC \subseteq M.Y \times \cup M.X$, internal coupling;
$SELECT : 2^M - \emptyset \rightarrow M$, tie-breaking function.

An overall system consists of a set of component models, either atomic or coupled, thus it is in a hierarchical structure. Each DEVS model, either atomic or coupled model, has correspondence to an object in a real-world system to be modeled. Within the DEVS framework, model design may be performed in a top-down fashion; model implementation in a bottom-up manner. Detailed descriptions for the definitions of the atomic and coupled DEVS can be found in [9].

3.3. DEVS Abstract Simulator

The *abstract simulator* introduced in [9] is a conceptual device capable of interpreting the dynamics of DEVS models. Two types of the abstract simulator have been defined: one for the *atomic DEVS* and the other for the *coupled DEVS*. Within the DEVSim++, a pair of a model and an abstract simulator is created, and association of the two is made for simulation. The job of an abstract simulator for an atomic model is to schedule the next event time and execute the model's transition functions and output function timely as simulation proceeds. The responsibilities of the abstract simulator for a coupled DEVS model are to synchronize the component abstract simulators for scheduling the next event time and to route external event messages to component simulators. The complete algorithms for both abstract simulators and proof for the correctness for the algorithms can be found in [10].

3.4. DEVSim++

DEVSim++ is an implementation of abstract simulator algorithm of DEVS based on C++[11]. It therefore provides the advantages of object-oriented framework, such as encapsulation, inheritance, and reuse. The DEVSim++ coordinates the event schedules of atomic models in a system and provides classes and APIs for simulation. DEVSim++ is provided as a software library to model developers, and synthesized with DEVS models, it forms a stand-alone simulator. The classical DEVS formalism deals with systems of closed form and static structure. There have been many researches to extend DEVS formalism to express variable structure, real-time and interactive systems. DEVSim++ supports these fea-

tures as well. DEVSim++ has an additional event queue to handle events from interactive components such as Graphical User Interfaces, and other monitor/controllers. These interactive components can be connected through networking or shared memory technology.

We get several advantages with the DEVS formalism and DEVSim++ to verify implementation of DEVS diagrams. First, we can specify the simulator models mathematically and easily verify the model. Second, we can model hierarchical and modularized systems, which enhance understandability and extensibility. Third, we can reuse simulation models.

4. DEVELOPMENT & EDUCATION PROCESS OF WARSHIP SIMULATOR PROJECT

In co-modeling methodology, M&S experts develop DEVS models that represent system behaviors in the state transition level. Domain experts are responsible for developing specific algorithms as individual functions. These algorithms typically require domain-specific knowledge. Majority of co-work between M&S experts and domain experts in co-modeling is to specify function prototypes. The functions should be designed independently and self contained as possible. DEVS models call associated functions inside each state transition function. The contents of the function, specifically the level of abstraction of equations used to implement the function, do not affect the model behavior as long as the I/O types of the function is consistent. Figure 2 describes these procedures.

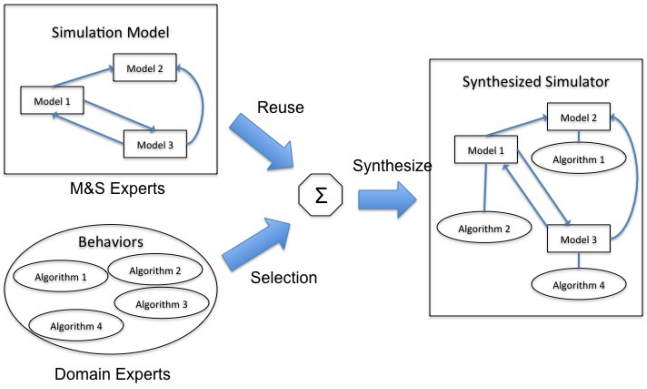

Figure 2. Developing Simulator using Co-modeling Methodology

During the Phase I of Warship Simulator Project, the M&S experts in SMSLAB developed the DEVS model based on the Simulation Logic Documents which domain experts in ADD wrote, and then domain experts and the M&S experts verified the DEVS models together. In Phase II, domain experts developed the DEVS models, and then the M&S experts verified the DEVS model based on the simulation aspect. In this section, we explain the phase of warship simulator development process and the results of the Warship Simulator Project in detail.

4.1. Phase I of the Warship Simulator Project

Since the M&S experts of SMSLAB do not have the domain knowledge, the domain experts in ADD made documents, called Simulation Logic Documents(SLD). In the SLD, it contains the use case diagram, sequence diagrams, and some functions to denote the simulation models. After receiving the documents, the M&S experts made the DEVS graph based on the sequence diagrams and the functions of the simulation models[2]. Figure 3 and Figure 4 are one of the sensor model of the sequence diagram of SLD and the DEVS model of it.

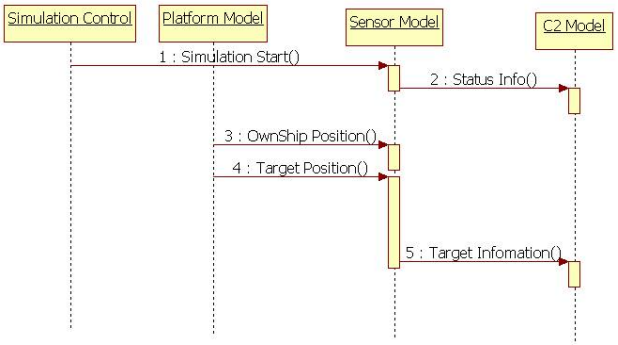

Figure 3. Sequence diagram of Sensor Model

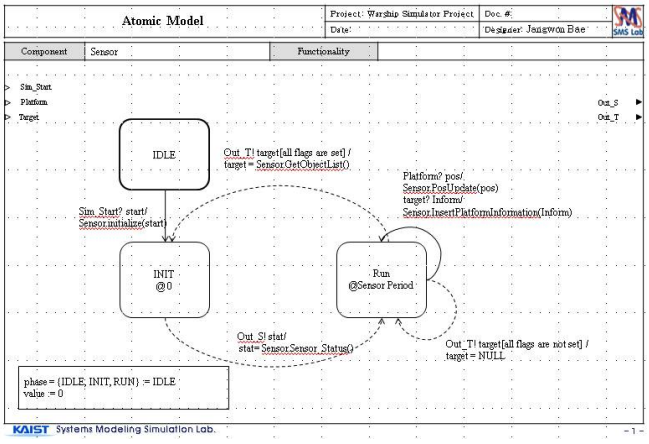

Figure 4. DEVS Graph of Sensor Model

After the SLD and DEVS graphs have been made, the domain experts and the M&S experts had a meeting to understand the SLD and DEVS graph. In order to assist the domain

experts to understand the DEVS graphs, we have made a prototype of the warship simulator using DEVSim++, and we have hand the source code to the ADD. By showing the results of the prototype in the debugging mode, the researcher of ADD have understood the simulation model easily. Figure 5 shows the prototype of the warship simulator.

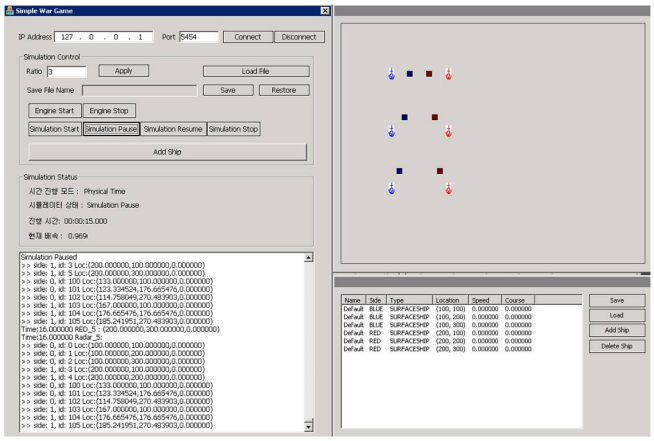

Figure 5. Prototype of warship simulator

After the meeting with ADD, we have found several misunderstanding of the simulation models. First, for some simulation models, the M&S experts have made the DEVS graph completely wrong due to the lack of domain knowledge. Second, the researchers have found some insufficient models based on the result of the prototype. Therefore, we get the revised version of SLD and the DEVS graphs modified based on the results of the meeting. According to these documents, ADD have implemented the object models and KAIST have implemented the DEVS models.

4.2. Phase II of the Warship Simulator Project

Based on the experience of Phase I, the ADD have developed their own simulation models in Phase II. During this Phase, The researcher of ADD tried to modify the source code of the prototype in Phase I, and test their models using the prototype before creating DEVS models. After that ADD have made serval DEVS models and they send the DEVS graphs to the KAIST. Especially, the researchers in ADD did not send the SLD due to the security polices. However, M&S experts of SMSLAB can analyzed the DEVS graph and tune the simulation models without modifying semantics of DEVS models. To verify a DEVS model, the information in SLD is not essential, but supplementary.

While the researchers of ADD build the DEVS model, the M&S experts of SMSLAB made the simulation framework using DEVSim++. This framework supports the simulator to change simulation model easily, i.e. Plug-in fashion. Therefore, it enables the the researchers of ADD to build their own models in DEVSim++ and analyze the result without violating security polices. Figure 6 is the overview of the framework.

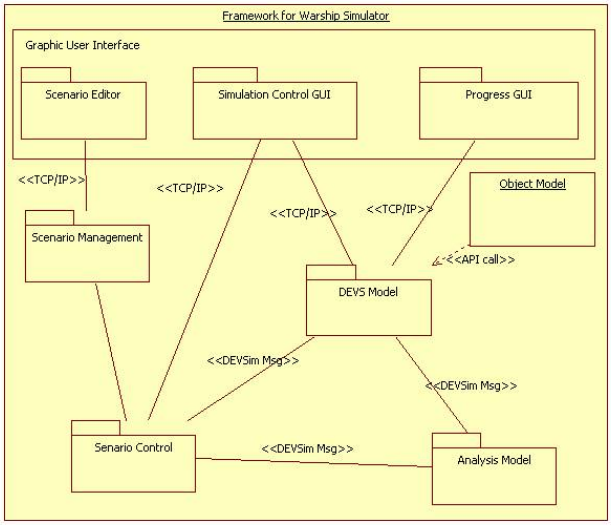

Figure 6. Plug-in Framework for warship simulator

5. LESSONS LEARNED

In this section, we share the lessons learned from Warship Simulator Project.

5.1. Necessity of formalism in development of a simulator

During the project, we are convinced that DEVS formalism is useful for the development of a simulator, and it is essential, not optional in many ways. We have applied the DEVS Formalism to develop a simulation models, and there are serval advantages which the researchers in ADD have agreed.

First, The DEVS formalism can support the modelers to make an accurate simulation models. By developing the DEVS graph, the domain experts can trust the simulation models. Moreover, by developing the DEVS models using DEVSim++, the modelers can verify their models quickly.

Second, The DEVS formalism are efficient to model the discrete event simulation model. In discrete event simulations, UML diagrams, e.g. sequence diagram, are not sufficient due to the properties of events. The events may be periodic and it may cause complex transitions based on the state of the models. Therefore, it is hard to express in the sequence diagram. However, the DEVS formalism specifies discrete event models in a hierarchical and modular form with Time information. Using these properties of DEVS formalism, complex events can be treated and complex models can be expressed by decomposing them into several models.

Third, the DEVS formalism is easy to learn, and it can be applied in the development of simulator easily. During the Phase I of the Warship Simulator Project, there were one lecture and two meetings for the researchers of ADD. After that, the researchers who took the lecture were able to teach and help other researchers to understand the DEVS formalism.

5.2. Prototyping & Education

We have found that the fast prototyping can assist the domain experts to understand the operations of the simulation models. Usually, the domain experts are familiar to programming languages so that the source code of simulator helped them to understand the operation of simulation model in the aspect of programming language. DEVSim++ is an implementation of abstract simulation algorithm of DEVS based on C++. Therefore, they were able to use integrated development environment, such as Visual Studio .NET, to follow the execution sequence, and it help them to understand the operations of simulation model quickly using the prototype.

In additions, the source code of prototype can assist the domain experts to understand the simulation model. During the Warship Simulator Project, we have handed the source code of warship simulator prototype to the ADD. The researchers of ADD have compared the source code and the design documents, i.e. DEVS graph, to know about the simulation models. Moreover, the domain experts have built their own models in C++ and test their models, and it enabled them to build accurate models without violating the security policies.

6. CONCLUSIONS

In this paper, we described our experience of developing the warship simulator and assisting the domain experts to make their own models by teaching M&S knowledge to them. During the Warship Simulator Project, we have found the formalism is necessary for the developing the simulators in military area, and DEVS formalism is suitable for simulator development. We believe that this case study can serve the domain experts how to obtain simulation knowledge and model the simulation model of their domains. As a future work, we will measure the effectiveness of DEVS formalism during the simulator development process formally.

REFERENCES

[1] R. Frigg and S. Hartmann, "Models in Science". Entry in the *Stanford Encyclopedia of Philosophy*, 2006.

[2] Su-Youn Hong and Tag Gon Kim, "Embedding UML Subset into Object-oriented DEVS Modeling Process," in *Proceedings of the Summer Computer Simulation Conference*, San Jose, California, USA, July 2004, pp. 161-166.

[3] Chang Ho Sung, Su-Youn Hong, and Tag Gon Kim, "Layered Approach to Development of OO War Game Models Using DEVS Framework," in *Proceedings of the Summer Computer simulation Conference*, 2005

[4] Jay K. Adams and Donald E. Thomas, "The Design of Mixed hardware/software systems," in *Proceedings of the 33rd annual conference on Design automation*, Las Vegas, Nevada, USA, June 1996, pp. 515-520.

[5] David W. Franke and Martin K. Purvis, "Hardware/Software CoDesign: A Perspective," in *Proceedings of the 13t international conference on Software engineering*, Austin, Texas, USA, 1991, pp. 344-352.

[6] Booch, Rumbaugh, and Jacobson. *The Unified Modeling Language User Guide*, Addison-Wesely, Reading, Massachusetts, 1998.

[7] Rumbaugh, *Unified Modeling Language Reference Manual*, Addison-Wesely, Reading, Massachusetts, 1999.

[8] Mark Priestley, *Practical Object-Oriented Design with UML*, The McGraw-Hill Companies, 1996.

[9] Bernard P. Zeigler, *Herbert Praehofer and Tag Gon Kim, Theory of Modelling and Simulation (2nd Edition)*, Academic Press; 2000.

[10] A.I. Concepcion and B.P. Zeigler, "DEVS Formalism: AFramework for Hierarchical Model Development," *IEEE Transactions on Software Engineering*, vol SE-14 no. 2, pp.228-241, Feb. 1988.

[11] Tag Gon Kim, *DEVSim++ User's Manual*, Available: http://smslab.kaist.ac.kr, 2007