Interoperation of DEVS Models and Differential Equation Models using HLA/RTI: Hybrid Simulation of Engineering and Engagement Level Models

Chang Ho Sung, Jeong Hee Hong and Tag Gon Kim
Department of EECS, KAIST
373-1 Guseong-dong, Yusong-gu
Daejeon, Korea, 305-701
{**chsung, jhhong**}**@smslab.kaist.ac.kr, tkim@ee.kaist.ac.kr**

Keywords: Hybrid Simulation, DEVS, MATLAB/Simulink, HLA/RTI, Combat System Effectiveness

Abstract

A hybrid model is a combination of models developed in various simulation development environments. A war game model is a kind of hybrid model and consists of multiple level models. Engineering level models are interested in the change of values via time and process mathematical equations, and most of them are developed in MALTAB/Simulink. Engagement level models are concerned about state transitions by events and modeled using the Discrete Event Systems Specification (DEVS) formalism. This paper proposes a framework for interoperation of simulation tools for engineering and engagement level models simulation. The framework employs the High Level Architecture (HLA) to reuse existing models and suggests specific interfaces for MATLAB/Simulink. The proposed framework is demonstrated through simulation of a simple war game model including heterogenous models.

1. INTRODUCTION

Systems modeling and simulation process is various according to the types and modeling objectives of the systems. Given modeling objectives, modelers should determine modeling levels and simulation tools. For example, continuous models are interested in the change of values via time and discrete event models are concerned about state transitions by events. On the other hand, hybrid models are heterogeneous models with continuous and discrete time components that act together to perform a function not possible with any one of individual system types, and each component is modeled in its own modeling tool.

A war game system is the representative example of hybrid systems and consists of various kinds of systems. Generally, modeling and simulation of a system in military service is divided into two levels: Engagement level is the upper level, and engineering level is the lower level [1]. Engagement level models are processed by external and internal events, and they are modeled as discrete event models. Engineering level models present detailed behavioral characteristics such as mathematical equations, physics and etc.

In this paper, DEVS formalism and DEVSim++ library are used for engagement level models and MATLAB/Simulink is used for engineering models. DEVS formalism is the methodology for modeling discrete event systems, and MATLAB/Simulink is a mathematical tool applied in various fields and used for modeling continuous systems.

There has been some effort for simulation of hybrid models. Most of the researches concentrate on specifying a unified methodology for hybrid simulation such as Ptolemy II [2], AnyLogic [3]. In addition, there was a research to simulate separate models of different levels using MATLAB/Simulink interface such as CODIS [4] and BCVTB [5]. The problems of previous researches are to be difficult to reuse existing models, and modelers have to insert specific interfaces into them. Thus, this paper proposes the interoperation of simulation of models of different levels to reduce the modification of existing models.

This paper is organized as follows. Section 2 reviews the DEVS formalism and MATLAB/Simulink as modeling and simulation environments. Section 3 explains the proposed framework for interoperation of simulation tools for different level models simulation. In section 4, a simple war game model is designed and simulated using the proposed framework. Conclusion is made in section 5.

2. MODELING AND SIMULATION ENVIRONMENTS

2.1. DEVS Formalsim

The DEVS formalism specifies discrete event models in a hierarchical and modular form. 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 [6].

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 states;
δ_{ext} : $Q \times X \rightarrow S$, an external transition function, where $Q = \{(s,e) | s \in S \text{ and } 0 \leq e \leq ta(s)\}$, total state set of M;
δ_{int} : $S \rightarrow S$, an internal transition function;
λ : $S \rightarrow Y$, an output function;
ta : $S \rightarrow R^{+}_{0,\infty}$ (non-negative real number), time advance function.

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 being in 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.

2.2. Simulation of DEVS Models

Simulation of a DEVS model can be done by communicating hierarchical abstract simulators the architecture of which is the same as the DEVS model architecture. The abstract simulators are a set of distributed simulation algorithms which can be implemented in a sequential as well as in a distributed computing environment [7]. Since DEVS models are developed in object-oriented manner, it is natural that an environment for DEVS modeling and simulation is implemented in an object-oriented programming language such as C++ or java. In fact, the first such C++ implementation is DEVSim++ [8] [9] in which modeling facilities and abstract simulators are explicitly separated. In the environment modeling facilities are opened to modelers; abstract simulators are not accessible externally. The facilities allow modelers to create DEVS models of type atomic or coupled, which are subclasses of the class atomic or coupled model defined within DEVSim++.

2.3. MATLAB/Simulink

2.3.1. MATLAB

MATLAB integrates mathematical computing, visualization, and a powerful language to provide a flexible environment for continuous systems modeling and simulation. A user can use MATLAB in a wide range of 252 applications, including signal and image processing, communications, control design, system modeling and analysis [10].

2.3.2. Simulink

Simulink is software for modeling and simulation that provides a graphical user interface for building models as block-diagrams [11]. It is designed to work in graphical environments such as Windows. Hence the most natural means of including information or models in Simulink is to draw them. Simulink has various functional blocks, and all of the tasks can be reduced by suitably linking these blocks. In the following sub-sections, two techniques to be used in this paper are described.

2.3.3. S-function

S-function (System-function) is a computer language description of Simulink. S-function can be written in MATLAB, C, C++ or Fortran. It uses a special calling syntax that enables a user to interact with Simulink equation solvers. By following a set of simple rules, a user can implement an algorithm in an S-function and use an S-function block to add it to a Simulink model. A MATLAB language, known as M-file, is used for S-function in this paper.

3. FRAMEWORK FOR HYBRID SIMULATION

3.1. Separation of Models

War game models comprise diverse models to satisfy varying simulation objectives. This paper proposes interoperation methodology for hybrid systems which consists of engineering and engagement level models. Models are divided into engineering and engagement level models according to simulation objectives as shown in Figure 1.

The objective of engagement level models, corresponding to modeling and simulation for development of combat systems, is analysis of doctrine, tactical evaluation and measure of effectiveness (MOE) [1]. Technically, engagement level models represent discrete event systems and employ DEVS formalism to do modeling.

Engineering level models are typically detailed mathematical representations of individual systems and modeled using MATLAB/Simulink. In addition, engineering level models are classified into two classes. The one conducts just algorithms without time information, and the other simulates the

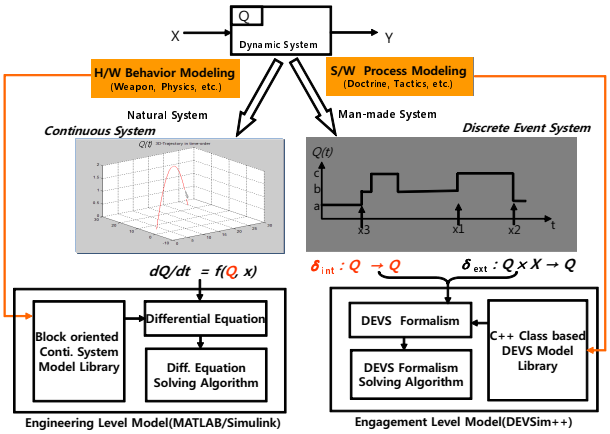

Figure 1. Modeling of dynamic systems according to modeling levels

change of states via continuous time. While the former employs a simple mathematical calculation, the latter has continuous variables affected by time, like ordinary differential equation. In fact, war game models contain various engineering level models. This paper considers all two types of engineering level models to propose the framework for interoperation.

3.2. Simulators Interoperation

3.2.1. HLA

The High Level Architecture (HLA) is a specification for interoperation among distributed heterogeneous simulations. The HLA specification is approved as IEEE 1516 standard in 2000. The standard consists of three pars: HLA Framework and Rules [12], Federate Interface Specification [13] and Object Model Template (OMT) [14].

HLA Framework and Rules defines HLA, its components, and the 10 rules that outline the responsibility of HLA federates and federations to ensure a consistent implementation.

Federate Interface Specification defines essential services of 6 categories. These services are also listed as a form of Application Programming Interfaces (APIs) written in various programming languages. The Run-Time Infrastructure (RTI) is an implementation of HLA Federate Interface Specification. The HLA only defines APIs and lefts detailed algorithms to RTI software developers.

OMT is a documentation format for shared objects and message types by which simulators exchange events. OMT is used to describe Federation Object Model (FOM), Simulation Object Model (SOM), Management Object Model (MOM). The FOM is a collection of all shared objects and interactions defined inside whole distributed simulation, i.e. federation. The SOM is a list of shared objects and interactions that a simulator, i.e. federate, sends to or receives from other simulators. Finally, the MOM supports control function to organize a federation, miscellaneous functions for synchronization points, save and restoration, monitoring and management functionality.

3.2.2. Instrument Control Toolbox

In order to exchange data between a federate and a MATLAB/Simulink model, we use the Instrument Control Toolbox supporting network services, such as TCP/IP, UDP and so on. The Instrument Control Toolbox enables to communicate with hardware and MATLAB, and provides commonly used communication protocols, such as GPIB [15][16], VISA, TCP/IP and UDP. This paper exploits TCP/IP interfaces for connecting to networked instruments and builds S-function blocks including the Instrument Control Toolbox functions for forwarding received data from external components to blocks within MATLAB/Simulink.

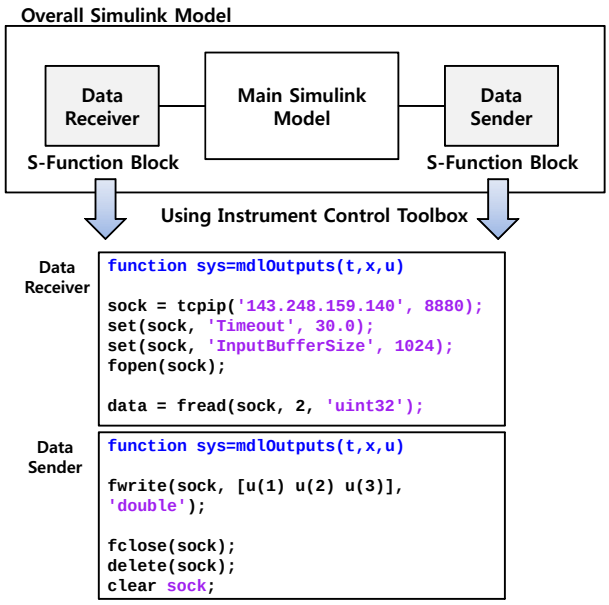

Figure 2. S-function block using the Instrument Control Toolbox

Figure 2 shows the construction of S-function blocks using the Instrument Control Toolbox. The original model for a continuous system is *Main Simulink Model* in Figure 2. We defines two S-funcion blocks and places them in the front and the rare of the original simulation model. The blocks are a *data receiver* and a *data sender*, and they communicate with external components and *Main Simulink Model*. After receiving data from external instruments, *data receiver* delivers the data into *Main Simulink model*. *Main Simulink model* manages principal operations using input data from *data receiver*, and transmits the processing data to *data sender* for sending data to external components. For sending or receiving data, it needs communication socket offered by the Instrument Con-

trol Toolbox. *Data receiver* exploits tcpip, fopen and fread inferfaces for creating, connecting socket and receiving data through socket, respectively. *Data sender* utilizes fwrite interface for sending data via socket. Detailed description of the functions are in Table 1.

Table 1. Functions for TCP/IP interface

Function	Description
tcpip	Create TCP/IP object
fopen	Connect interface object to instrument
fclose	Disconnect interface object from instrument
fread	Read binary data from instrument
fwrite	Write binary data to instrument

3.2.3. Framework for Simulators Interoperation

Engagement and Engineering level models are implemented by DEVSim++ and MATLAB/Simulink, respectively. Interoperation among simulators operates under the HLA/RTI environment mentioned in Section 3.2.1., and we use a KHLAAdaptor library [17]. The KHLAAdaptor library ,which offers all HLA services as well as abstract callback services, enables developers to be easily able to build interoperable simulators for interoperation. Interoperable simulators include HLA modules and stand-alone simulators. HLA modules manage mapping between HLA services and simulation messages for simulators. Simulators are able to reuse existing simulators developed by DEVSim++ or MATLAB/Simulink. The TCP/IP protocol is utilized in order for communication between simulators and HLA modules. Figure 3 shows framework for interoperation between DEVSim++ and MATLAB/Simulink.

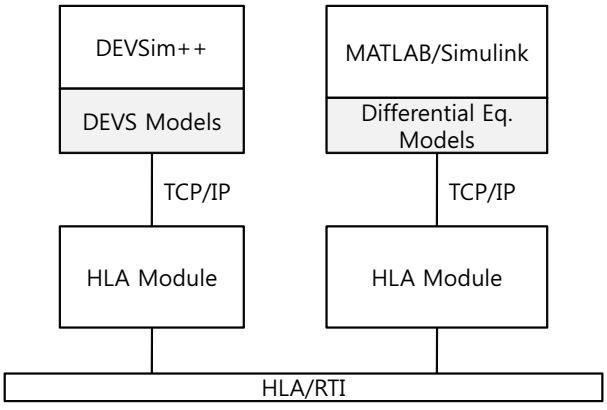

Figure 3. Framework for simulators interoperation

Figure 4 describes time advancement mechanism for interoperation between discrete event and continuous systems. MATLAB/Simulink represented the continuous system successively computes operations of the system at intervals from simulation start time to finish time. The successive time points at which operations are executed are called time steps. At each time step, MATLAB/Simulink computes only one operation. On the other hand, event time of discrete event systems modeled using DEVS formalism is aperiodic due to event-driven system. Each system demands time advancement corresponding next step from RTI. The next step of the continuous system may be next time step while discrete event system may be next event time. RTI grants the minimum time advancement among requested time. Thus, the role of time management module of RTI allows whether continuous or discrete event systems to advance their time to minimum among requested time.

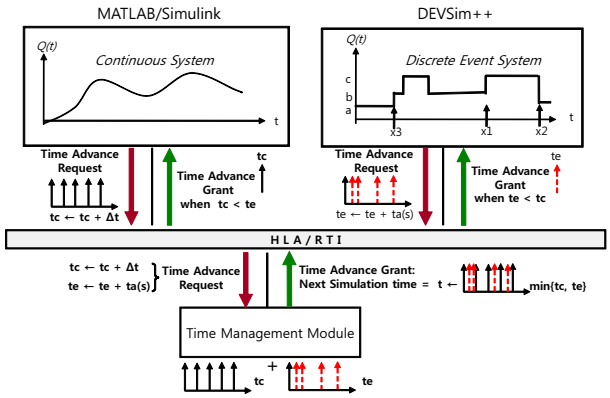

Figure 4. Time Advancement Mechanism for interoperation

4. CASE STUDY

The proposed framework is applied to simulation of simple war game models depicted in Figure 5. The objective of the simulation is to calculate the survivability of a warship. The simulator is useful to increase the survivability of the ship by testing various parameters of defense systems.

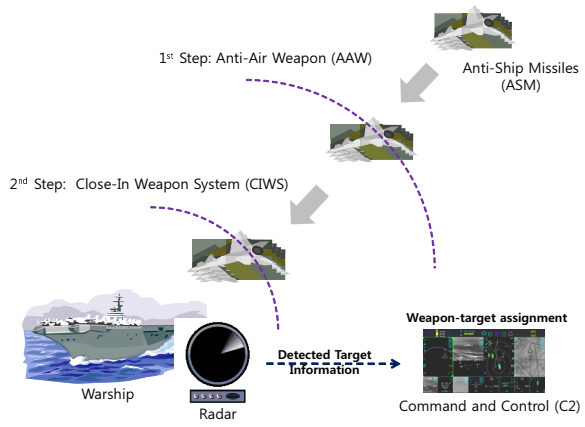

Figure 5. Defense of ship against anti-ship missiles

The brief scenario of the war game simulation is as follows.

1. A warship is moving to a certain destination point.
2. A radar is detecting any hostile missiles toward the ship.
3. A warship has two steps of defense systems: Anti-Air Weapon (AAW) and Close-In Weapon System (CIWS). The systems have different effective ranges and are activated by command and control (C2) system.
4. If an ASM hits the warship or it is shot down, the simulation ends.

The war game simulation models have both engagement and engineering modeling level. The example model consists of four models. The models and their roles are as follows.

- Engagement Level Model
  1. Warship Model
     As a core model of others, it controls overall simulation of a war game model. It sends initial variables and states to other models. This model takes an important role of maneuver and engagement analysis. When the warship model receives weapon information from the C2 model, it changes the state and analyzes the engagement results. This model is specified using the DEVS formalism because it is activated by state transition with some events.
- Engineering Level Model
  1. Radar Model
     It detects positions of a warship and a ASM periodically, and sends the information to the C2 model. Mathematical functions of MATLAB/Simulink are used to decide whether the ASM is in the range of detection.
  2. ASM Model
     It calculates next position using following movement equations by periods and sends the position information to the radar model.

$$x = v_x t + x_0$$

$$y = v_z t + y_0$$

$$z = \int_0^t (v_z + gt)dt + z_0, \;\; g = -9.8 m/s^2$$

  3. C2 Model
     It determines whether a warship can attack an ASM or not. When the C2 model gets position information from a radar model, it assigns the proper weapon according to two steps of defense systems: AAW or CIWS. AAW is a kind of missile shot at long range, and CIWS is a weapon for destroying ASMs at short range. Decision model is described in MATLAB/Simulink, and it has no timing information and processes only static algorithms.

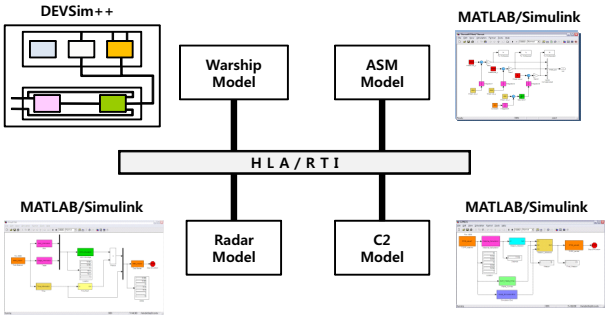

Figure 6. Overall simulation architecture of a war game model

Overall simulation architecture of the war game model is described in Figure 6. The war game model consists of one DEVS model and three MATLAB/Simulink models, and all of the models have HLA modules as shown in 3.2.3.. MATLAB/Simulink models include the Instrument Control Toolbox for TCP/IP as described in 3.2.2.. The purpose of the simulation is to analyze survivability of a warship according to detection range of a radar model. The simulation is activated with following parameters of Table 2, and a warship stays at initial position while an ASM moves against the warship using specific movement equations. Detection range changes at an interval of 5km from 10km to 35km. Figure 7 shows the simulation result of the war game model. In the longer detection range of a radar, the probability of a warship's survival is higher.

Table 2. Simulation parameters

Parameter	Value
Total simulation count	100
# of warships	1
Initial position of warships $(x,y,z)km$	(0,0,0)
# of targets	1
Initial position of targets $(x,y,z)km$	(20,20,0)
# of AAWs	6
# of CIWSs	100
Range of CIWSs (km)	≤ 5

5. CONCLUSIONS

This paper proposes a framework for interoperation of simulation tools to simulate both engineering and engagement level models. The one represents a discrete event system, and the other does a continuous system. Each model is developed separately using MATLAB/Simulink and DEVSim++,

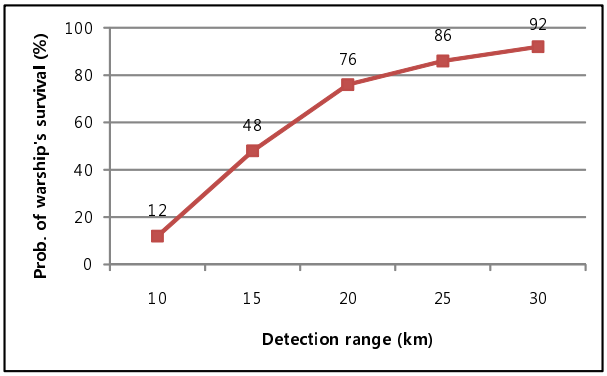

Figure 7. Survivability of a warship

and Instrument Control Toolbox and HLA are used for interoperation between models. For simulation of disparate models, we should insert interface codes using Instrument Control Toolbox and develop adaptors of models without modification of existing models. A case study demonstrates co-simulation among engineering and engagement level models. Further studies are remained for complete reusability of existing simulation models.

6. ACKNOWLEDGEMENT

This work was supported by Defense Acquisition Program Administration and Agency for Defense Development under the contract UD080042AD, Korea.

REFERENCES

[1] Robert M. McGraw and Richard A. MacDonald, "Abstract Modeling for Engineering and Engagement Level Simulations," in *2000 Winter Simulation Conference*, 2000, pp. 326 – 334.

[2] Haiyang Zheng, "Operational semantics of hybrid systems," Ph.D. dissertation, Berkeley, CA, USA, 2007, adviser-Edward A. Lee.

[3] Andrei Borshchev, Yuri Karpov, and Vladimir Kharitonov, "Distributed simulation of hybrid systems with AnyLogic and HLA," *Future Gener. Comput. Syst.*, vol. 18, no. 6, pp. 829–839, 2002.

[4] Faouzi Bouchhima, Gabriela Nicolescu, El Mostapha Aboulhamid, and Mohamed Abid, "Generic discrete-continuous simulation model for accurate validation in heterogeneous systems design," *Microelectron. J.*, vol. 38, no. 6-7, pp. 805–815, 2007.

[5] Michael Wetter and Philip Haves, "A Modular Building Controls Virtual Test Bed for The Integration of Heterogeneous Systems," in *3rd National Conference of IBPSA-USA*, 2008, pp. 69 – 76.

[6] Bernard P. Zeigler, Tag Gon Kim, and Herbert Praehofer, *Theory of Modeling and Simulation*. Orlando, FL, USA: Academic Press, Inc., 2000.

[7] Bernard P. Zeigler, *Multifacetted modelling and discrete event simulation*. San Diego, CA, USA: Academic Press Professional, Inc., 1984.

[8] Tag Gon Kim and Sung Bong Park, "The DEVS Formalism:Hierarchical Modular Systems Specification in C++," June 1992, pp. 152 – 156.

[9] Tag Gon Kim, *DEVSimHLA User's Manual*, 2007. [Online]. Available: http://smslab.kaist.ac.kr

[10] MathWorks, *Using MATLAB Manual*. The MathWorks Inc., 2007.

[11] MathWorks, *Using SIMULINK Manual*. The MathWorks Inc., 2007.

[12] "IEEE standard for modeling and simulation (M&S) high level architecture (HLA) - framework and rules," *IEEE Std 1516-2000*, pp. i–22, Sep 2000.

[13] "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification," *IEEE Std 1516.1-2000*, pp. i–467, 2001.

[14] "IEEE standard for modeling and simulation (M&S) high level architecture (HLA)-object model template (OMT) specification," *IEEE Std 1516.2-2000*, pp. i–130, 2001.

[15] "IEEE Standard Digital Interface for Programmable Instrumentation," *IEEE Std 488.1-1987*, Jun 1988.

[16] "IEEE Standard Codes, Formats, Protocols, and Common Commands for Use with IEEE Std 488.1-1987, IEEE Standard Digital Interface for Programmable Instrumentation," *IEEE Std 488.2-1992*, Dec. 1992.

[17] Jae-Hyun Kim, Su-Youn Hong, and Tag Gon Kim, "Design and Implementation of Simulators Interoperation Layer for DEVS Simulator," in *M&S-MTSA'06*, July 2006, pp. 195 – 199.