Realization of the DEVS Formalism
In MATLAB/Simulink

Kyung M. Seo*, Tag G. Kim**, and Chang H. Sung***
School of EECS KAIST
373-1 Kusong-dong, Yusong-gu
Daejeon, Korea 305-701
Tel: +82-42-869-5454
Fax: +82-42-869-8054
E-mail: kmseo@smslab.kaist.ac.kr*, tkim@smslab.kaist.ac.kr**, chsung@smslab.kaist.ac.kr***

Keywords: DEVS formalism, MATLAB/Simulink, discrete event simulation

Abstract

The DEVS (Discrete Event Systems Specification) formalism supports specification of discrete event models in a hierarchical modular manner. MATLAB/Simulink is a widely used tool for modeling, simulating and analyzing continuous and discrete event systems. This paper proposes a realization of the DEVS formalism in MATLAB/Simulink. The proposed design enables to use a great amount of mathematical packages and functions included in MATLAB/Simulink. The design is also employed as real time simulation and hybrid system simulation and a hybrid system is a mixture of continuous systems and discrete event systems. The paper introduces Simulink-DEVS model, in which a simulation algorithm is embedded. The model consists of a Simulink-atomic model and a Simulink-coupled model. In addition, the time advance algorithm to simulate the model is suggested. The algorithm handles the time synchronization and the accommodation of different concepts specific to continuous and discrete event models. Two experimental results are presented for a pure discrete event model and a hybrid model.

1. INTRODUCTION

The DEVS (Discrete Event Systems Specification) formalism is widely employed for Modeling and Simulation (M&S) of discrete event systems. It is used to build DEVS models for performance evaluation and logical analysis. Recently, the DEVS formalism has become used to model complex systems such as computer/communication networks and manufacturing systems [1]. For this reason, the analysis and design of DEVS models require technical computing tools. Among the various kinds of tools, MATLAB/Simulink is a popular tool in all engineering fields. It integrates mathematical computing, visualization, and several programming languages, such as C++ and Fortran, to provide a flexible environment [2]. In addition, it is familiar to M&S engineers to model and simulate discrete event systems as well as continuous systems.

The purpose of this paper is the integration of the DEVS formalism into MATLAB/Simulink. Central to this paper are two ideas: The first is the Simulink-DEVS model, the realization of the DEVS formalism in MATLAB/Simulink, and the second is the time advance algorithm to simulate the model. Figure 1 describes the overview of the proposed framework.

![Figure 1 Overview of the Proposed Framework](image)

The proposed design and implementation utilize a various collection of visualizations and computations using MATLAB/Simulink. Moreover, it enables the co-existence of discrete event and discrete time systems in a uniform simulation environment. Especially, a system that contains both continuous and discrete event system is defined as a hybrid system [4].

There is similar research regarding discrete event and hybrid systems simulation in MATLAB/Simulink. A simulation product using SimEvents, which is embedded in Simulink, has been presented to model and simulate discrete event and hybrid systems [3]. However, the product is limited to consideration of discrete event systems using queues and servers. Furthermore, this technique is difficult to apply the formalism of a discrete event system, such as the DEVS formalism. Therefore, it is hard to manipulate discrete event systems mathematically.

This paper is organized as follows: Sections 2 and 3
present a brief review of the DEVS formalism and MATLAB/Simulink, respectively. Section 4 exploits a Simulink-DEVS model, and Section 5 introduces a time advance algorithm. Section 6 shows the possibilities of the Simulink-DEVS model with experimental results. Finally, Section 7 concludes this research and proposes future extensions for a more complete solution.

2. DEVS FORMALISM: BRIEFD REVIEW

2.1. DEVS Formalism

The DEVS formalism, a set-theoretic formalism, specifies discrete event systems in a hierarchical and modular form. With the formalism, one can specify a discrete event system more easily by decomposing a large system into smaller component models. The DEVS formalism consists of two kinds of models: Atomic model and Coupled model [5].

An Atomic model is the basic models and has specifications for the dynamics of the model. Formally, a 7-tuple specifies an Atomic model \( M \) as follows:

\[
M = < X, Y, S, \delta_{\text{ext}}, \delta_{\text{int}}, \lambda, ta >,
\]

where

- \( X \): a set of input;
- \( Y \): a set of output events;
- \( S \): a set of sequential states;
- \( \delta_{\text{ext}} : Q \times X \rightarrow S \), an external transition function,
  where \( Q = \{(s,e) | s \in S, 0 \leq e \leq ta(s)\} \) is the total state set of \( M \);
- \( \delta_{\text{int}} : S \rightarrow S \), an internal transition function;
- \( \lambda : S \rightarrow Y \), an output function;
- \( ta : S \rightarrow \text{Real} \), time advance function.

A Coupled model provides the method of assembly of several Atomic and/or Coupled models to build complex systems hierarchy. Formally, a Coupled model is defined as follows:

\[
DN = < X, Y, M, EIC, EOC, IC >,
\]

where

- \( X \): a set of input events;
- \( Y \): a set of output events;
- \( M \): a set of all component models;
- \( EIC \subseteq DN.X \cup M.X \) : external input coupling;
- \( EOC \subseteq \cup M.Y \times DN.Y \) : external output coupling;
- \( IC \subseteq \cup M.Y \times \cup M.X \) : internal coupling;
- \( \text{SELECT} : 2^M - \phi \rightarrow M \) : tie-breaking selector.

An overall system consists of a set of component models, either Atomic or Coupled, thus being in a hierarchical structure. Each DEVS model, either Atomic or Coupled model, has correspondence to an object in a real-world systems 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. DEVS Abstract Simulator

The DEVS Abstract simulator concept associated with the DEVS formalism characterizes what has to be done to execute atomic or coupled models with hierarchical structure. The simulation agents exchange simulation messages in Table 1 along the hierarchy in order to run simulation. The simulator agent for an atomic model is called a simulator. The simulator agent for a coupled model is called a coordinator [5].

Table 1 DEVS Simulation Message

<table>
<thead>
<tr>
<th>Message</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>(( \ast ), t)</td>
<td>Internally generated event at time that notifies the scheduled time is completely elapsed</td>
</tr>
<tr>
<td>(x, t)</td>
<td>External input event at time t</td>
</tr>
<tr>
<td>(y, t)</td>
<td>Internal output event at time t</td>
</tr>
<tr>
<td>(done, ( \ast ))</td>
<td>Synchronization event generated at time t that notifies the next scheduled time is ( t + \ast )</td>
</tr>
</tbody>
</table>

Figures 2 and 3 show the DEVS abstract simulator algorithm and the DEVS abstract coordinator algorithm, respectively.

3. MATLAB/SIMULINK

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
applications, including signal and image processing, communications, control design, system modeling and analysis [2].

3.2. Simulink

Simulink is software for M&S that provides a graphical user interface for building models as block-diagrams [7]. 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.

3.2.1. S-Function

An 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 a 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.

3.2.2. Subsystem

A Subsystem block represents a subsystem of the system that contains it. As the model increases in size and complexity, it can be simplified by grouping blocks into a subsystem. A Subsystem enables a user to establish a hierarchical block diagram, where a Subsystem block is located on one layer and the blocks that make up the subsystem are on another. A Subsystem is created in this way: The blocks are enclosed and lines that the user wants to include in the subsystem are connected within a bounding box. In this paper, a Subsystem block is used for modeling Simulink-coupled model.

4. SIMULINK-DEV'S MODEL

A Simulink-DEV'S model is the implementation of the DEV'S formalism in MATLAB/Simulink. It consists of a Simulink-atomic model and a Simulink-coupled model, which correspond to a DEV'S atomic model and a coupled model, respectively. In particular, a Simulink-atomic model includes a simulation algorithm for this type of the model.

4.1. Simulink-atomic Model

Figure 4 illustrates a DEV'S atomic model and a Simulink-atomic model. The Simulink-atomic model provides the templates of three set blocks, four function blocks and a simulation algorithm block. Four function blocks and a Simulation Algorithm block are implemented by M-file S-Function, as mentioned earlier. A user implements four function blocks and three set blocks according to the template of a Simulink-atomic model. It is unnecessary for the user to consider the simulation algorithm block. A set of input events and output events is implemented using an In port block and an Out port block in Simulink. State sets are declared as global variables in four function blocks. Subsequently, function blocks in a Simulink-atomic model can access the states without the connection among the blocks. The simulation algorithm block, which corresponds to the abstract simulator algorithm, and an Initialization block will be described in detail, in Section 4.3.

Simulink is based on a discrete time simulation. Accordingly, this paper assumes that the time advance value must be positive. Therefore, negative or zero time advance values are not considered in this paper.

4.2. Simulink-coupled Model

The Simulink-coupled model is implemented using Subsystem, which represents a virtual subsystem of a system in Simulink. Figure 5 shows the relationship between a coupled model and a Simulink-coupled model.
As the model M12 in Figure 5 is opened, the window displays the blocks, M1 and M2, which are included in M12. In port and Out port blocks reflect signals entering and leaving M12. Assuming that a Simulink-coupled model M3 has two In port blocks, and two input events through In port blocks are received at the same time, the Simulink solver cannot decide which message to handle, first. Therefore, the select function in the DEVS formalism is beyond the scope of this paper. The Simulink-coupled model takes a primary role of proper translation of messages between connected models. The model is achieved using Subsystem automatically. Hence, contrast with the Simulink-atomic model, Simulink-coupled model is unnecessary to have any extra simulation algorithm block.

4.3. Simulation Algorithm of a Simulink-atomic Model
This Section describes the simulation algorithm of a Simulink-atomic model. A simulator algorithm, as presented previous Section 2.2, is represented as a Simulation Algorithm block in Simulink-atomic model. Figure 7 shows the structure of a Simulink-atomic model. At each Simulink clock, all blocks in Simulink-atomic model are executed by the determined order, next Simulink clock is updated. The Simulink-atomic model includes a Simulation-Algorithm block and an Initialization block. The Simulation-Algorithm block takes a role of simulator algorithm in DEVS abstract simulator algorithm. The Initialization block decides the first event time and the initial state of this Simulink-atomic model. Figure 6 shows the pseudo code of a Simulation-Algorithm block. An event message, (i, t) is for initialization to get the first event time and the initial state of the Simulink-atomic model.

![Figure 6 Pseudo Code of a simulation algorithm](image)

Figure 7 The Structure of a Simulink-atomic Model

5. TIME ADVANCE ALGORITHM IN SIMULINK
To execute a simulation using the proposed Simulink-DEVS model, the mapping between the Simulink clock and the simulation time is required. The Simulink clock is a time step that computes new values for inputs, states, and outputs of the model. It also updates the computed values. The Simulation Time Generator translates the clock into the simulation time. One of the most difficult problems in implementing the Simulation Time Generator is that it must allow a proper time advance of the simulation, regardless of the type of the systems. In order to solve this problem, the Simulink Time Generator was designed to support either an event-driven simulation or a time-driven simulation according to the type of systems. Figure 8 shows the role of the Simulation Time Generator block, and Figure 9 illustrates a pseudo code used in Simulation Time Generator.

![Figure 8 Simulation Time Generator](image)

![Figure 9 Pseudo Code of Simulation Time Generator](image)
For an event-driven simulation, the Simulation Time Generator should report all next event times of the model. At each Simulink clock, Simulation Time Generator receives \((\text{done}, t_S)\) of all models and calculates the minimum next event time, \(t_S\). An event \((*, t_S)\) is delivered to all Simulink-atomic models. Each Simulink-atomic model receives \((*, t_S)\), and follows the directions of the simulation algorithm. When Simulation Time Generator takes a part of the discrete time simulation, Simulation Time Generator transmits \((*, t_S)\) where \(t_S\) is a pure Simulink clock.

6. EXAMPLES

To demonstrate the efficiency of the proposed Simulink-DEVS model and time advance algorithm, two experiments were conducted with the framework, explained in Section 1. The first experiment involves a single server queuing model, and shows the performance evaluation. The second experiment involves a war game model. It is a hybrid model and shows hybrid system simulation in a uniform simulation environment.

6.1. Single Server Queuing Model

The single server queuing model consists of four Simulink-DEVS models: Generator, Queue, Server, and Transducer. Figure 10 illustrates the architecture of the single server model. The bottom picture in Figure 10 shows the Simulink-atomic model architecture.

![Figure 10 Single Server Queuing Model](image)

Figure 10 shows the simulation results for two different simulation environments. DEVSim++ realizes the DEVS formalism for modeling and associating abstract simulator concepts for simulation, in C++ [9]. Server’s service time is the duration of service per one entity [10]. The server’s service time is the normal distribution with standard deviation 1. Figure 11 depicts the average queue length in accordance with server’s service time. In the case of three distributions of the server’s service time, the average queue length is the same in both simulations.

![Figure 11 Simulation Result](image)

6.2. War Game Model

The war game model calculates survivability of a warship, and to evaluate performance of AAW (Anti-Air Warfare) and CIWS (Close-In Weapon System) systems in the ship against an ASM (Anti-Ship Missile) [11]. Figure 12 shows the architecture of the war game model.

![Figure 12 War Game Model](image)

When the warship detects the approaching ASM, it acts two types of defense systems, AAW and CIWS. The war
game model is a hybrid model, composed of Simulink-DEVS models and continuous models. Engage, Analysis and Target models are Simulink-DEVS models, and Radar and CFCS (Command Firing and Control System) models are continuous models. The simulation is applied to the real time simulation. Figure 13 is the result for the naval engagement model. Power of Target, the performance measurement in this example, is the strength of the missile to approach toward the warship. As simulation time increases, the distance between the warship and the missile is decreased, and the power of the missile is decreased.

7. CONCLUSION
This paper presents the design and implementation of the DEVS formalism in MATLAB/Simulink. A Simulink-DEVS model and a flexible time advance algorithm are proposed. A Simulink-DEVS model is the implementation of the DEVS formalism in MATLAB/Simulink, and a time advance algorithm supports conversion between a Simulink clock and a next event time. Using this framework, developers can build a Simulink-DEVS model easily by implementing the templates of the Simulink-DEVS model. This technique enables real time simulation, as well as hybrid system simulation. Two examples show the effectiveness of the proposed modeling methodology. Implementation of the Simulink-DEVS model as a library form should be considered in a future work.

8. REFERENCES