REAL-TIME SIMULATION FRAMEWORK BASED ON RT-DEVS FORMALISM

Seong Myun Cho and Tag Gon Kim

Systems Modeling Simulation Lab.,
Department of Electrical Engineering & Computer Science, *KAIST*
373-1 Kusong-dong, Yusong-Ku, Taejon 305-701, KOREA
e-mail: smcho{tkim}@smslab.kaist.ac.kr

ABSTRACT

Real-time, discrete event simulation often employs a simulation model that interacts with a surrounding environment, such as software components, hardware components or human operators. In such simulation, a simulator must interact with its environment such that events from its environment should be processed in a timeliness manner. Simulation methodologies developed for non-interactive simulation, such as performance simulation, do not have such capabilities in event processing. We propose a real-time, interactive simulation methodology which concurrently executes real-time DEVS(RT-DEVS) models and its environment in a time selective manner. Real-time execution environment is implemented to execute RT-DEVS models based on the proposed simulation methodology.

Keywords: Real-time simulation, RT-DEVS formalism, Real-time scheduling

1. INTRODUCTION

Discrete-event simulation is widely used not only to predict performance but also to analyze behavior, of complex systems. Real-time, discrete event simulation is a means to verify a real-time system in which a simulation model interacts with a surrounding environment, such as software components, hardware components or human operators. In such simulation, a simulator must handle external events from its environment in a timeliness manner. Simulation methodologies developed in non-interactive simulation, such as performance simulation, do not have such capabilities in event processing. This is because such methodologies employ virtual time, not real-time, in event scheduling and processing. We propose a real-time, interactive simulation methodology for combined discrete event models and a surrounding environment. Within the methodology, a real time system is modeled by the RT-DEVS formalism, a real-time extension of Zeigler's DEVS formalism[Zei84]; interactive components are used as they are. The methodology concurrently executes real-time DEVS (RT-DEVS) models and its environment in a time selective manner. Real-time execution environment is implemented to execute RT-DEVS models based on the proposed simulation methodology.

This paper is organized as follows. Section 2 describes the RT-DEVS formalism used in modeling of real-time subsystems. The proposed scheduling policy is given in section 3. An algorithm for real-time simulation and its realization are given in section 4. Conclusion is given in section 5.

2. RT-DEVS FORMALISM

The real-time DEVS formalism is an extension of the DEVS formalism for real-time systems simulation. An

atomic model in RT-DEVS formalism, *RTAM*, is defined as:

Real-Time Atomic Model:

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

, where

X : a set of external input event types
S : a sequential state set
Y : an output set
$\delta_{ext.} : Q \times X \rightarrow S$, an external transition function,
where Q is the total state set of $M = \{(s,e) \mid s \in S \text{ and } 0 \leq e \leq ta(s)\}$
$\delta_{int.} : S \rightarrow S$, an internal transition function
$\lambda : S \rightarrow Y$, an output function
$ta : S \rightarrow I^{+}_{0,\infty} \times I^{+}_{0,\infty}$, a time interval function,
where $I^{+}_{0,\infty}$ is the non-negative integer numbers with ∞ adjoined. Note that a time advance, *ta(s)*, for RT-DEVS is given by an interval $ta(s)|_{\min} \leq ta(s) \leq ta(s)|_{\max}$.
$\psi : S \rightarrow A$, an activity mapping function
A : a set of activities $A = \{a \mid t(a) \in I^{+}_{0\infty} \text{ and } t(a) \leq ta\big|_{max}\} \cup \emptyset$

In the DEVS formalism[Zei84], virtual simulation time advances only when a simulator calls the time advance function *ta* of an atomic model. The RT-DEVS formalism replaces virtual time advance by real time advance. The actual advance of simulation time is sum of the real execution time of activity function and the time residing in blocked state of the corresponding atomic model. There is one restriction of the specification of an atomic model. Atomic model cannot receive events when its state has an activity function.

A real-time DEVS coupled model connects basic real-time DEVS models together in order to form a new model. As in the DEVS formalism, a set of component models makes up a new coupled model. The real-time coupled DEVS model is defined as:

Real-Time Coupled Model:

$$RTCM = < D, \{M_i\}, \{I_i\}, \{Z_{i,j}\} >$$

, where

D, I_i, Z_i : same as the original
M_i : a component basic real-time DEVS model

A coupled model within the RT-DEVS formalism is defined the same way as in the original DEVS formalism[Zei84] with an exception. The exception is that there is no *SELECT* function in RT-DEVS, which has been defined in the DEVS formalism to break ties for simultaneous events scheduling. This is because such simultaneous events can not be happened in a real-time simulation environment. In real-time simulation with single processor, only one event at a time can be physically processed even if more than one event occurred from the external environment.

3. SCHEDULING ALGORITHM

3.1 EVENT DRIVEN SCHEDULING

To meet timing requirement under real-time simulation, the execution priority of a model should be determined by its time advance value. In RT-DEVS formalism, the next scheduling time of a model is associated

with t_N value, which is given as time advance value at last scheduling time. One important observation is that RT-DEVS models have varying t_N value during execution. Therefore, their execution priority should be given dynamically and an atomic model with smaller t_N value must be given higher priority. Besides, atomic models are supposed to update their t_N value whenever they receive internal or external event. When a model receives an event, the real-time model scheduler sets the execution priority of that model to highest one. This is because the corresponding model needs to make urgent rescheduling, which updates t_N value. Therefore, the cause of priority reassignment is events that the corresponding model receives. That is why we refer our scheduling algorithm to *event driven scheduling algorithm*. Table 3.1 summarizes the characteristics of the event driven scheduling algorithm.

	Event Driven Scheduling
Task Characteristic	*RTAM* $= < X, S, Y, \delta_{ext}, \delta_{int}, \lambda, ta, \psi, A >$ Aperiodic event driven task
Goal of Scheduling	Preservation of Causality
Factor of Priority Assignment	Updated scheduling time due to received event and event itself
Dependencies among tasks	Input/output dependencies influences execution orders

Table 3.1 Characteristics of event driven scheduling

4. SIMULATION ENVIRONMENT REALIZATION

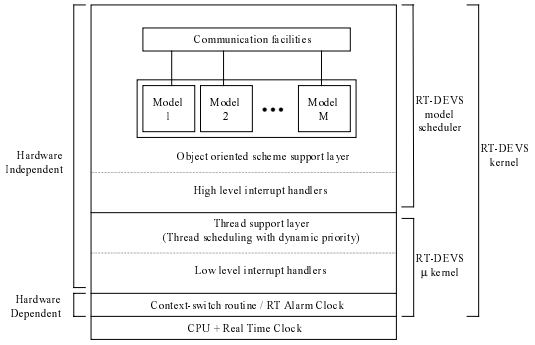

Figure 4.1. The architecture of RT-DEVS simulation environment

4.1 ARCHITECTURE

The architecture of RT-DEVS simulation environment is shown in Figure 4.1. The RT-DEVS micro kernel provides basic functions that enables model threads to be executed concurrently. RT-DEVS micro kernel is composed of two layers. At lower layer, there exist thread context-switch routine and real-time alarm clock management routine. Real-time alarm clock routine supplies facilities, which produce current time and support timer services. Thread support layer enables the CPU to switch between model threads and interrupt routines handles events from external environment.

The primary goal of our simulation framework is to run RT-DEVS models interacting with its environment on

time. In other words, the simulation framework should guarantee for all simulation models to evolve properly on physical time. To achieve this goal, we made real-time model scheduler, which executes each RT-DEVS models as real-time tasks by using facilities that the thread support layer provides. The RT-DEVS model scheduler changes priorities of simulation models based on the proposed priority assignment principle. While on the contrary the RT-DEVS micro kernel can be viewed as a kernel-thread execution engine, the real-time model scheduler can be viewed as a model execution engine.

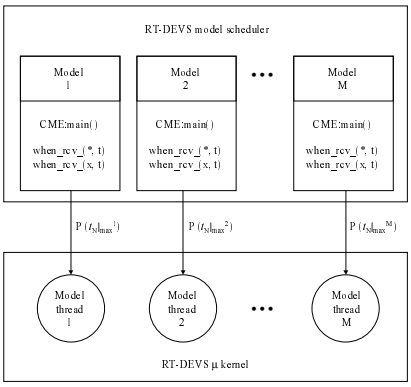

Figure 4.2 Relation between real-time model scheduler and RT-DEVS micro kernel

Initially modeler specifies the target system in RT-DEVS formalism and makes simulation models, which are RT-DEVS atomic/coupled models. Within the environment, coupled models are flattened to minimize communication delays during real-time simulation. Each atomic model spawns its own thread to execute its characteristic functions concurrently with other model threads associated with corresponding atomic models. The real-time model scheduler performs CME(Concurrent Model Execution**)** algorithm, where it not only executes characteristic functions of each model but also changes the execution priority of each model. This is done by calling the priority function P with argument $t_N|_{max}$ of each model and a model with smaller $t_N|_{max}$ is given higher priority. Then, a model with highest priority is executed, for underlying kernel-thread engine always switches CPU to the model thread with highest priority. This mechanism is depicted in Figure 4.2.

4.2 CME (Concurrent Model Execution) ALGORITHM

The execution of an atomic model is done by the following two algorithms:

1. **CME:when_rcv_(*, t):**
2. **if** $t_N|_{min} \le t \le t_N|_{max}$ **then**
3. y := λ(s);
4. send message(y, t) to associated ports;
5. s := δ_{int} (s);
6. $t_L := t$;
7. $t_N := [t_L + ta(s)|_{min}, t_L + ta(s)|_{max}]$;
8. $P(t_N|_{max})$
9. ψ(s)
10. **else**
11. error;
12. **end if**

Figure 4.3 CME: when_rcv(*, t)

The above algorithm **when_rcv_(*, t)** handles an internal event (*, t) which notifies that a scheduled time t comes. The CME performs this procedure whenever an internal time out occurs without any interruption of external events. First, the CME checks the validity of internal timeout. Lines 3 to 7 of the algorithm correspond to the function calls sequence [λ, δ_{int}, ta]. After updating its time advance value, it modifies its execution priority based on that by calling priority function P, which also updates priorities of other simulation models based on current t_L value. RT-DEVS model with smaller $dt_N (= t_N|_{max} - t_L)$ value is given higher execution priority.

1. **CME:when_rcv_(x, t):**
2. **if** $t_L \le t \le t_N|_{max}$ **then**
3. $e := t - t_L$;
4. $s := \delta_{ext}(s,e,x)$;
5. $t_L := t$;
6. $t_N := [t_L + ta(s)|_{min}, t_L + ta(s)|_{max}]$;
7. $P(t_N|_{max})$
8. $\psi(s)$
9. **else**
10. error;
11. **end if**

Figure 4.4 CME: when_rcv(x, t)

The above algorithm **when_rcv_(x, t)** handles an external event (x, t). The CME performs this procedure whenever an external event is received. As in the previous algorithm, it first checks the validity of the external event's generation time. Lines 3 to 6 of the algorithm correspond to the function calls sequence [δ_{ext}, ta]. As before, it modifies its execution priority based on updated dt_N value by calling priority function P.

Whenever the priority function is called, the kernel-thread engine switches CPU to the model thread with highest priority.(Line 8 in Figure 4.3 and Line 7 in Figure 4.4) A context switching is also occurred, when a model performs output function.(Line 4 in Figure 4.3) This is because a model that has a pending external event is given highest priority. Therefore, a model with pending external event gets the CPU.

1. **CME:main():**
2. $s = s_0$ /* initialize */
3. $t_N := [ta(s_0)|_{min}, ta(s_0)|_{max}]$;
4. **concur_forever for each RT-DEVS model** /* main loop */
5. wait for a event
6. **if** an external event **then**
7. when_rcv_(x, t);
8. **else if** an internal time out event **then**
9. when_rcv_(*, t);
10. **end if**
11. **end concur_forever**

Figure 4.5 CME: main()

The above algorithm **main()** shows the main routine of the CME. Each atomic model waits for an external event until an internal timeout occurs. If an external event is occurred before an internal timeout expires, it

proceeds **when_rcv_(x,t)**. If an internal time expires without external event, it proceeds **when_rcv_(*,t)** and sets new internal timeout.

To synchronize time between atomic models and environment, The model scheduler suspends atomic model after finishing its activity until the time specified by its time advance function. If an external event occurs during that interval, the scheduler wakes up the corresponding atomic model.

4.3 REAL-TIME SIMULATION STRATEGY

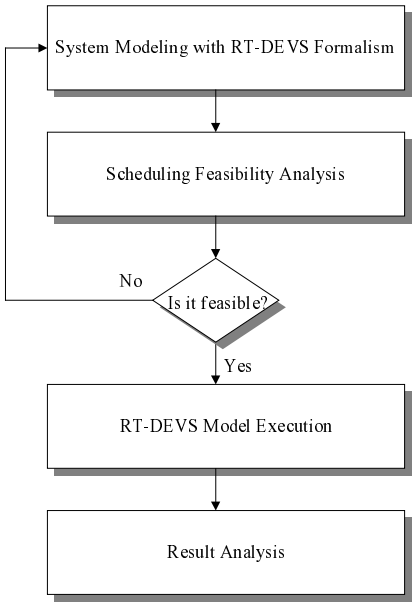

Figure 4.6 Simulation procedure

A typical simulation procedure under our real-time simulation environment is depicted in Figure 4.6. First, the target system is modeled with RT-DEVS formalism. Modeler specifies atomic/coupled models and couplings between them, which represents the target system. Schedulability test graph is extracted from it and corresponding transient graphs are generated. Then, the checking of the scheduling feasibility is performed on each transient graph. If the given models are not acceptable for real-time simulation, modeler reconstructs RT-DEVS models to make them acceptable for real-time simulation. If this is the case, modeler often divides time consuming activity function into smaller ones and modifies corresponding atomic models to make it suitable for the proposed real-time scheduling.

When the given simulation models pass the scheduling feasibility analysis, the real-time model scheduler now simulates them. First, the real-time model scheduler flattens all coupled modes to minimize communication delays between atomic models. After flattening of all coupled models, it spawns model threads for corresponding atomic models and establishes communication channels between them. When real-time simulation is done without any run-time exception due to timing violation, modeler can analyze the simulation result.

5. CONCLUSION

We presented a real-time simulation methodology by extending the DEVS formalism and employing concurrent execution of RT-DEVS models. The real-time simulation framework employs priority based scheduling

to meet the timing requirement given by RT-DEVS models. Therefore, we also implemented real-time execution environments.

The RT-DEVS simulation methodology and its execution environment provide the uniform and flexible representation of real-time systems.

References

[CK98] Seong Myun Cho and Tag Gon Kim, "Real-Time DEVS Simulation: Concurrent, Time-Selective Execution of Combined RT-DEVS Model and Interactive Environment." *The proceedings of the summer computer simulation conference*, July 1998.

[CP97] Alberto Coen-Porisini, Carlo Ghezzi and Richard A. Kemmerer, "Specification of real-time systems using ASTRAL." *IEEE transactions on software engineering*, vol. 23, no. 9, 1997.

[JHong97] Joon Sung Hong, Hae Sang Song, Tag Gon Kim and Kyu Ho Park, "A Real-Time Discrete Event System Specification Formalism for Seamless Real-Time Software Development." *Discrete Event Dynamic Systems: Theory and Applications*, 7, 355-375, 1997.

[Kim91] Tag Gon Kim, Hierarchical development of model classes in the DEVS-Scheme simulation environment. *Expert Systems With Applications*, 3:343-351, 1991.

[KS97] C.M. Krishna and Kang G. Shin, *REAL-TIME SYSTEMS.* McGraw-Hill Book, Oxford, 1994.

[MO94] Leo Motus and Michael G. Rodd, *Timing analysis of real-time software.* Redwood Books, Oxford, 1994.

[Par93] Sung Bong Park, DEVSIM++: A semantic based tool for object-oriented modeling of discrete event systems. Master's thesis, Dept of Electrical Eng., KAIST, 1993.

[RAMA94] Ramamrithan, K., and Stankovic, J. "Scheduling Algorithms and Operating Systems Support for Real-Time Systems." *Proceedings of the IEEE*, January 1994.

[RP97] Alan Burns and Andy Wellings. *Real-Time Systems and Programming Languages.* Addison-Wesley, 1994.

[Zei84] Bernard P. Zeigler, *Multifacetted Modelling and Discrete-Event Simulation.* Academic Press, New York, 1984.

[ZK93] Bernard P. Zeigler and Jinwoo Kim, "Extending the DEVS-scheme knowledge-based simulation environment for real-time event-based control." *IEEE Transactions on Robotics and Automation*, vol. 9, no. 3, 1993