ANALYSIS OF FEASIBILITY FOR REAL TIME SIMULATION OF RT-DEVS MODELS

SEONG MYUN CHO and TAG GON KIM

Systems Modeling Simulation Laboratory
Dept. of Electrical Engineering & Computer Science, KAIST
373-1 Guseong-dong Yuseong-Gu, Daejeon 305-701, KOREA

Abstract

Real time simulation denotes a simulation that has interactions with a surrounding environment, such as software components, hardware components or human operators. In such simulation, a proper scheduling policy and its feasibility analysis method are needed to ensure that the timing requirements are satisfied, which inherently depend on the underlying computational models.

This paper proposes a scheduling policy and its feasibility analysis method for the real time DEVS(RT-DEVS) models. The RT-DEVS model provides features such as object-orientation, state description of behaviors, and formal semantics for executability. The proposed analysis method consists of two phases. The one is the state synchronization analysis to see if the model specifications are consistent with their interactions. The other is the feasibility analysis for real time simulation under the proposed scheduling policy.

Keywords

Real time simulation, scheduling feasibility, real time DEVS formalism.

1 Introduction

During the construction of real time simulation models, it is vital to ensure that timing requirements are satisfied by the system under development. Satisfying the timing constraints is very difficult due to the lack of explicit specification of such constraints in the program.

To solve this problem, real time objects have been proposed[AS91][ITM90][KK94][MT90]. A real time object is a basic object that has a set of operations with timing constraints and threads, each of which is an execution unit. The main reason to define the object models is to encapsulate not only data but also timing. Mercer[MT90] presented the object model used in the ARTS real time kernel. Ishikawa[ITM90] proposed RTC++, a real time extension of C++, and defined a real time object RTO which has two types of threads, slave and master. Attoui[AS91] proposed another real time object model MO2. MO2 is an object-oriented model which integrates the features of both DBMS and real time systems. Kim and Kopetz[KK94] suggested RTO.k (also called time-triggered RT-object) model. RTO.k extends the conventional object models in four ways: deadline for each execution of a method, separation of time-triggered methods and message-triggered methods, maximum validity duration of real time data, and basic concurrency constraint. They suggested execution engine model of the DREAM kernel and implemented them[KK94].

In these methods, a modeler can specify timing requirements in model specification. However, they lack sound formalism in model specification, thus remaining a timing analysis as a difficult problem. A typical example of a timing analysis is the feasibility analysis, which is about whether the target system models can meet their timing requirements under the given scheduling policy.

We present a feasibility analysis method for RT-DEVS models under the event driven scheduling policy. In the absence of deterministic arrival of events, it is hard to determine response times, and thus hard to analyze a system's behavior. Therefore, in order to improve tractability of formal analysis, we decided to sacrifice the generality by assuming the periodic state trajectory.

This paper is organized as follows. Section 2 describes the RT-DEVS formalism used in modeling of real time subsystems. The feasibility analysis method is discussed in detail in section 3. Conclusion is given in section 4.

2 Real Time DEVS formalism

The real time DEVS formalism is an extension of the DEVS formalism[Zei84] for real time systems simulation and is used in modeling system in our simulation environment. An atomic model in RT-DEVS formalism[JHong97], RTAM, is defined as

Definition 2.1 (Real Time Atomic Model)

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

where

X: a set of input event
S: a sequential state set
Y: a set of output event
δ_{ext}: $Q \times X \rightarrow S$,
an external transition function where
Q is the total state set of
$M = \{(s,e) | s \in S \text{ and } 0 \le e \le ta(s)|_{max}\}$
δ_{int}: $S \rightarrow S$, an internal transition function
λ: an output function
ta: $S \rightarrow I^{+}_{0,\infty} \times I^{+}_{0,\infty}$, a time interval function
Note that a time advance, ta(s), for RT-DEVS is given by an interval $ta(s)|_{min} \le ta(s) \le ta(s)|_{max}$, $s \in S$
ψ: $S \rightarrow A$, an activity mapping function
A: a set of activities,
$A = \{a | t(a) \in I^{+}_{0,\infty} \text{ and } t(a) \le ta|_{max}\} \cup \emptyset$

The RT-DEVS formalism replaces virtual time advance in the DEVS formalism[Zei84] by real time advance. Note that the RT-DEVS formalism defines a set of activities associated with a state, which is not defined in the DEVS formalism. Each activity is an abstraction of a task in a real system to be modeled. Note also that a time advance, ta(s), for RT-DEVS is given by an interval $ta(s)|_{min} \le ta(s) \le ta(s)|_{max}$ of integers, which is a real number in DEVS. A reason for such interval time advance is that a RT-DEVS simulator checks a specified time advance of a RT-DEVS model against a real time clock within the simulator during simulation. $ta(s)|_{min}$ is an auxiliary parameter which is used in the verification of timing correctness during simulation. The analysis method in the later section deals with $ta(s)|_{max}$ only, for the run time priority of a model is related with its current $ta(s)|_{max}$ under the proposed scheduling policy. If the clock falls in the scheduled time interval, correctness of the schedule would be verified. Integers in time advance specify computation time of an activity in terms of ticks generated by an underlying computing system.

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. A real time DEVS coupled model in RT-DEVS formalism, RTCM, is defined as:

Definition 2.2 (Real Time Coupled Model)
$RTCM = \langle D, \{M_i\}, \{I_i\}, \{Z_{i,j}\} \rangle$
where
D: a set of component names
For each i in D
M_i: a component basic real-time DEVS model
I_i: a set of influences of i
and for each j in I_i
$Z_{i,j}$: $Y_i \rightarrow X_j$, an i-to-j output translation

A RTCM consists of components M_i, which are atomic models and/or coupled models. The influences I_i and i-to-j output translations $Z_{i,j}$ define the coupling specification(CS) as follows: An external input coupling(EIC) connects the input event of the coupled model to the input event of one of its components; An external output coupling(EOC) connects the output event of a component to the output event of the coupled model; An internal coupling(IC) connects the output event of a component to the input event of another component.

3 Real Time Simulation Feasibility

This section presents the *feasibility analysis for real time simulation*(FRTS) under the proposed scheduling policy. The two-phase analysis procedure will be discussed in detail. First, the proposed scheduling policy is explained.

3.1 Event Driven Scheduling

By prioritizing the execution order of simulation models, we can make the state transitions of the RT-DEVS models occur on time. Such prioritizing is needed to prevent unacceptable delay, thus violating the timing requirement of every simulation model. In the RT-DEVS formalism, the next scheduling time of a model is associated with $t_N (= ta(s)|_{max})$, which is given as the time advance value at the latest scheduling time. A RT-DEVS model with smaller t_N has higher priority and events are processed with the highest priority under our scheduling policy. One important observation is that t_N of a RT-DEVS model is changing during the simulation execution. More specifically, atomic models are supposed to update their t_N whenever they receive internal or external events. When a model receives an event, the real time model scheduler sets the execution priority of that model to the highest one. This is because the corresponding model needs to make urgent rescheduling, which updates t_N. Therefore, priority reassignment is made whenever a model receives an event. That is why we refer our scheduling algorithm to as *event driven scheduling algorithm*.

3.2 Basic Assumptions

In this section, we describe our basic assumptions. For the feasibility analysis, the following assumptions on the RT-DEVS models are made.

1. *Invariant Structure*: The structure of an overall RT-DEVS model is not variable during simulation.
2. *Periodic State Trajectory*: The composed system model has a periodic state trajectory.
3. *Basic Concurrency Restriction*: A model doesn't receive events when its current state executes an activity function associated with it.

The second assumption above is essential to analyze the feasibility for real time simulation of the RT-DEVS models, which we define as follows.

Definition 3.1 (Periodic State Trajectory)
Consider a system model composed of model m^1, m^2,..., m^m with a global states set of $S = \{ s_1, s_2, ..., s_n \}$. For each state $s_i \in S$ of the system, there is a corresponding sojourn time $s.t_i$ that represents the duration for which the system stays in that state. Assume that the system performs a deterministic sequence of state transitions $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow ... \rightarrow s_n$. Then, the system is said to have a periodic state trajectory if it repeats such a state sequence periodically.

Generally speaking, a RTAM could have infinite number of state trajectories depending on the input sequences given to it. However, the assumption of periodic state trajectory selects a specific state sequence among them, which is determined by the context of the target system. In the proposed analysis method, the assumed state sequence is verified from the model specification. Then, it is used in the analysis of the feasibility for real time simulation.

3.3 Proposed Analysis Methodology: Two-Phase Approach

The procedure of the feasibility analysis is depicted in Figure 1. To begin with, all RTCMs, except the target system model which is a RTCM, are flattened into RTAMs before the analysis. The analysis procedure divides into two phases. The one is the analysis of the state synchronization among RTAMs and the other is the analysis of the feasibility for real time simulation(FRTS).

The state synchronization analysis is to see if the model specifications are consistent with their interactions. For the state synchronization analysis, the state trajectory of a RTAM is defined. The state trajectory of RTAM is its expected state sequence in the target system, which the modeler has in mind. If a model has interactions with other models, there are unknown sojourn times in states where the model is expecting an event from the other model. All the sojourn times of a model must be determined to specify its state sequence completely. We formulate a set of simultaneous equations that represent the state synchronizations among the component models. If the solutions of these equations have all positive values, it denotes that the interactions among the models are consistent with the model specifications. If the system model fails in the state synchronization analysis, the target system is modeled again to make the interactions be consistent with the model specifications. This process will be illustrated in next section.

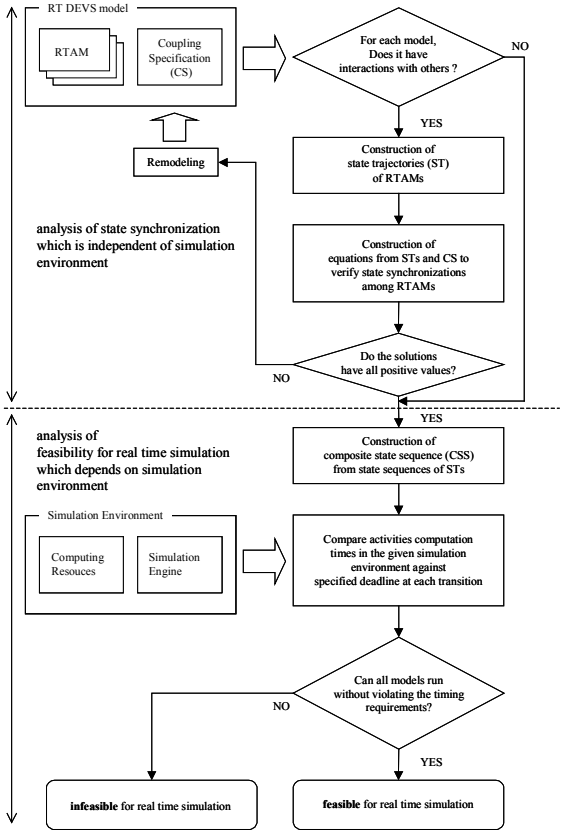

Figure 1. Two-Phase Analysis of RT-DEVS models

After state trajectories of all component models are completely specified, through the incremental composition of them, the global state sequence of the target system is constructed. For the composition of state sequences, the composite state sequence is defined. The global state sequence represents the target system's behavior. Therefore, it can be used to verify the FRTS. It can be verified by applying the FRTS steps, which will appear in later section. In the FRTS, the computation times of activity functions are obtained from the simulation environment.

The FRTS steps checks all activity functions can be finished on time by comparing the computation times in the given simulation environment against the

specified deadline at each transition. As stated earlier, we assume that the composite target system model has a periodic state trajectory. Therefore, the analysis over a period suffices for the determination of the feasibility for real time simulation.

3.3.1 State Synchronization Analysis

The state trajectory is an expected timed state sequence of a RTAM in a target system. The formal definition of state trajectory is as follows.

Definition 3.2 (State Trajectory of RTAM)

$$ST = < X, Y, S, \{s.t_i\}, \sigma, \Gamma, \Theta>$$

where
X, Y, S: Same as those in the RTAM
$\{s.t_i\}$: Sojourn time set
σ: Timed state sequence
Γ: $S \rightarrow X$, State-input mapping function
Θ: $Y \rightarrow S$, Output-state mapping function
with constraints

$s.t_i$, sojourn time in s_i of m
$\sigma \subseteq \{(t_i,s_i)\}$ an ordered set with a relation $\leq$ on t_i
$t_0 = 0,\ t_i = \sum_{k=0}^{i-1} s.t_k$ where $i \geq 1$
$s_i \in S$, i is the index in σ,
s_0 is the initial state

Note that Γ gives the input event x which the model m is expecting in the given state s, while Θ gives the state s in which the model m generates the given output event y.

The sojourn time isn't necessarily equal to $ta(s)|_{max}$ given in the specification of RTAM, for a RT-DEVS model may not be executed alone. Instead, a set of such models is executed together with interactions among them. To specify the state trajectory of RTAM completely, we must determine unknown sojourn times in the state sequences given by the assumption. This process is explained as follows.

Sojourn time:
Let $s.t(m:s_i)$ be the sojourn time in a state s_i of a model m, where subscript i denotes the sequence index and s_0 the initial state.

For states with no external input event:
In this case, the sojourn time is obtained directly from the model specification.

$$s.t(m:s_i) = ta(s_i)|_{max} \text{ of m}$$

For states with an external input event:
In this case, there must be a model that sends the event to this model, which is denoted as m'. When m' in the state s_j sends an event to m in the state s_i, both models synchronize their state transitions. Therefore, the sojourn time can be expressed in the following form.

$$\sum_{k=0}^{i-1} s.t(m:s_k) + s.t(m:s_i) = \sum_{k=0}^{j} s.t(m':s_k)$$

This equation is called a state synchronization equation. Among its items, the sojourn times in states with no external event are substituted with their values in the model specification. After all sojourn times of models are expressed in the above form, a set of simultaneous equations is obtained. Then, the sojourn times are determined by solving these equations.

The algorithm depicted in Figure 2 describes the formulation of state synchronization equations. Let ST^m denote the state trajectory of model m. Line 4-6 in Figure 2 is to find the synchronized state for a state with an external event by the use of Γ, Θ in ST and CS in the model specification. After solving the equations obtained, all unknown sojourn times are determined. If all the sojourn times have positive values, the models' interactions are consistent with their specifications. Note that, in Line 7 and 8, some of $s.t(m:s_k)$ and $s.t(m':s_k)$ are substituted with their values in the model specification, if they are sojourn times in states with no external event.

Input: STs of the component models and CS of the target system model
Output: A set of state synchronization equations

1. **foreach** $m \in \{M_i\}$ of the target system model **do**
2. **foreach** s_i in σ of ST^m **do**
3. **if** $ta(s_i)|_{max} = \infty$ **then**
4. $x \leftarrow \Gamma(s_i)$ of ST^m
5. Find m' such that $m'.y \rightarrow m.x$ from CS
6. $s_j \leftarrow \Theta(y)$ of $ST^{m'}$
7. $\Delta \leftarrow \sum_{k=0}^{i-1} s.t(m:s_k) + s.t(m:s_i)$
8. $\Delta' \leftarrow \sum_{k=0}^{j} s.t(m':s_k)$
9. Form a state synchronization equation $\Delta = \Delta'$
10. **end if**
11. **end foreach**
12. **end foreach**

Figure 2. Formulation of Synchronization Equations

3.3.2 Feasibility Analysis

To analyze the feasibility for real time simulation of an overall RT-DEVS model, a means to represent a global state trajectory is to be devised. We solve this problem by composition of state sequences and use of the feasibility test graph

obtained from it. The composition of state sequences is defined as follows.

Definition 3.3 (Composite State Sequence)

$$\sigma^{m||n} = \sigma^m||\sigma^n$$

where

$\sigma^m||\sigma^n = \Lambda(\sigma^m|\sigma^n) = \{(t_i, s_i)\}$,

an ordered set with a relation $\leq$ on t_i,

$s_i \in S^m \times S^n$

$\sigma^m|\sigma^n$, the ordered set union of σ^m and σ^n

with constraints

Λ: Composition rules, where $s \neq s'$, $t < t'$

$\sigma^m|\sigma^n : (t, s^m), (t, s^n)$

$\rightarrow \quad \sigma^m||\sigma^n : (t, (s^m, s^n))$

$\sigma^m|\sigma^n : (t, s^m), (t, s^n), (t', s'^n)$

$\rightarrow \quad \sigma^m||\sigma^n : (t, (s^m, s^n)), (t', (s^m, s'^n))$

$\sigma^m|\sigma^n : (t, s^m), (t, s^n), (t', s'^m)$

$\rightarrow \quad \sigma^m||\sigma^n : (t, (s^m, s^n)), (t', (s'^m, s^n))$

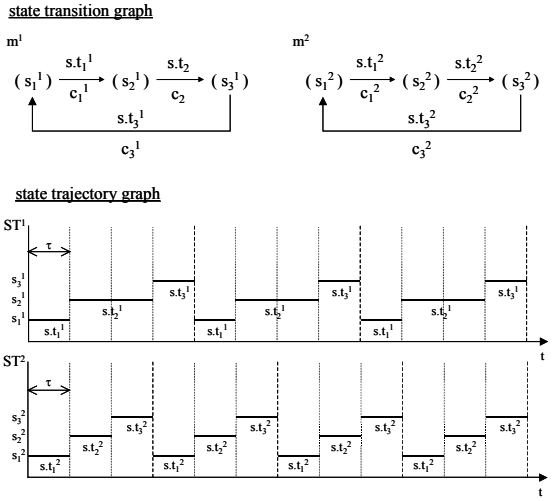

Figure 3. Model m^1 and m^2 with Periodic State Trajectory

To explain the composite state sequence of a composite model, consider two atomic RT-DEVS models m^1 and m^2 as shown in Figure 3, each having a periodic state trajectory. To show the timing relations between the two models, timed state trajectory graph is also depicted in Figure 3. Assume that m^1 and m^2 are executed concurrently. Then, the concurrent behavior of the two models can be represented by the composition of their state sequences.

$$\sigma^{1||2} = \sigma^1 \parallel \sigma^2$$

Let $T^1 = \Sigma s.t_i^1$ and $T^2 = \Sigma s.t_i^2$ and $T = LCM(T^1, T^2)$. Then, $\sigma^{1||2}$ is state sequence with period T, where LCM denotes *least common multiple*. Let σ^i be the state sequence of m^i up to T and $\tau = 1$, then

$\sigma^1 = \{ (0, s_1^1), (1, s_2^1), (3, s_3^1), (4, s_1^1), (5, s_2^1), (7, s_3^1), (8, s_1^1), (9, s_2^1), (11, s_3^1), (12, s_1^1)) \}$

$\sigma^2 = \{ (0, s_1^2), (1, s_2^2), (2, s_3^2), (3, s_1^2), (4, s_2^2), (5, s_3^2), (6, s_1^2), (7, s_2^2), (8, s_3^2), (9, s_1^2), (10, s_2^2), (11, s_3^2), (12, s_1^2) \}$

Let $\sigma^1|\sigma^2$ be the ordered set union of σ^1 and σ^2. (Recall that $\sigma = \{(t_i, s_i)\}$ is an ordered set with a relation $\leq$ on t_i.)

$\sigma^1|\sigma^2 = \{ (0, s_1^1), (0, s_1^2), (1, s_2^1), (1, s_2^2), (2, s_3^2), (3, s_3^1), (3, s_1^2), (4, s_1^1), (4, s_2^2), (5, s_2^1), (5, s_3^2), (6, s_1^2), (7, s_3^1), (7, s_2^2), (8, s_1^1), (8, s_3^2), (9, s_2^1), (9, s_1^2), (10, s_2^2), (11, s_3^1), (11, s_3^2), (12, s_1^1), (12, s_1^2) \}$

The composite state sequence $\sigma^{1||2}$ ($= \sigma^1||\sigma^2$) is obtained from $\Lambda(\sigma^1|\sigma^2)$.

$\sigma^1||\sigma^2 = \{ (0, (s_1^1, s_1^2)), (1, (s_2^1, s_2^2)), (2, (s_2^1, s_3^2)), (3, (s_3^1, s_1^2)), (4, (s_1^1, s_2^2)), (5, (s_2^1, s_3^2)), (6, (s_2^1, s_1^2)), (7, (s_3^1, s_2^2)), (8, (s_1^1, s_3^2)), (9, (s_2^1, s_1^2)), (10, (s_2^1, s_2^2)), (11, (s_3^1, s_3^2)), (12, (s_1^1, s_1^2)) \}$

Since the sequence of transitions represents concurrent behavior of the two on the time base, it can be used as a means to analyze timing properties of the composed system. Figure 4 shows the timed state sequence graph of the composite model consisting of m^1 and m^2 in Figure 3. Because all models have a periodic state trajectory, the graph of the composed timed state sequence also has a periodic form.

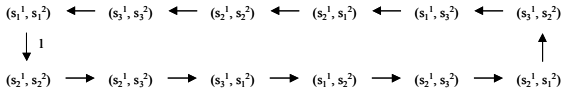

Figure 4. State Sequence of Composite Model

The global state sequence of the target system is constructed by incremental composition of the state sequences of the component models. It can be used to check its feasibility for real time simulation. From the global state sequence of the target system, we construct the FRTS graph. As depicted in Figure 5, each global state becomes a node. The edge between two consecutive nodes denotes the sojourn time which is tagged as its weight.

To determine whether the system models can fulfill activity functions without violating the timing requirements under the event driven scheduling

policy, the following steps are performed over the FRTS graph depicted in Figure 5. Note that c_n^m denotes the computation time of an activity function $\psi(s_n)$ of a model m under the simulation environment.

Step 1: For each node n, where $n \geq 1$, find the model m that made its state transition. i.e., $s_{n-1}^m \rightarrow s_n^m$, where $s_{n-1}^m \neq s_n^m$. This step is to identify models that have just passed its intermediate deadline.

Step 2: Find the node p in which the model m made its previous transition. i.e., the earliest adjacent node with s_{n-1}^m. In edges between this node and the current node, the activity function of the state s_{n-1}^m must be fulfilled.

Step 3: For edges between the two nodes p and n, in ascending order, decrement the edge weight by the amount of computation time c_{n-1}^m. Note that for all edges except the last one, the maximum amount of weight decrementable is the amount of the current edge weight. if the current edge weight is smaller than the computation time, it becomes zero and the next edge weight is decremented by the amount of the remainder computation time. This step is to see if there is available time to fulfill the activity function.

After performing the above steps, for all edges, if its weight is non-negative, then the target system is feasible for real time simulation. This condition is another way of stating that the system's utilization demand does not exceed 1. (This would render the models infeasible for real time simulation under any circumstances.)

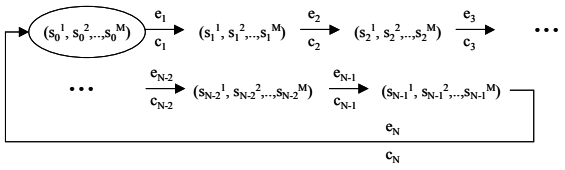

Figure 5. FRTS Graph

4 Conclusions

The RT-DEVS formalism provides a uniform and flexible description of real time systems. We have presented our experience with the application of feasibility analysis to RT-DEVS models under the event driven scheduling policy.

Our proposed analysis method divides into two phases. First, in the state synchronization analysis, state trajectory graphs are constructed which show the expected state sequences of models. This process involves determination of sojourn times in states of all models, where interactions among simulation models must be considered. In the feasibility analysis, by composition of the state sequences of the models that comprise the target system, the global system state sequence is obtained. By performing the FRTS steps on it, we can analyze the feasibility for real time simulation of the simulation models. The global system state sequence also can be used for other requirement analysis.

In order to improve tractability of formal analysis, we decided to sacrifice the generality by assuming the periodic state trajectory. Consequently, only simulation models with deterministic behavior in event time can be analyzed for the feasibility of real time simulation. Analysis of non-deterministic event time models is much harder and remains as a future work.

References

[AS91] A. Attoui and M. Schenider, An object oriented model for parallel and reactive systems, Proceedings of IEEE CS 12th Real time Systems Symposium, 1991, pp 84-93.

[ITM90] Yutaka Ishikawa, Hideyuki Tokuda and Clifford Mercer, Object-oriented real time language design: Constructs for timing constraints, Technical Report CMU-CS-90-111, Carnegie Mellon University, 1990.

[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, 1997, pp 355-375.

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

[KK94] K. H. Kim and Hermann Kopetz, A real time object model RTO.k and an experimental investigation for its potentials, International Computer Software and Applications Conferences, November 1994, pp 392-402.

[MT90] Clifford W. Mercer and Hydeyuki Tokuda, The ARTS real time object model, Proceedings of IEEE CS 11th Real time Systems Symposium, 1990, pp 2-10.

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