한국시뮬레이션학회 '00추계학술대회 논문집 2000.11.11 아주대학교

실시간 시뮬레이션을 위한 스케줄 가능성 분석 기법

조 성 면, 김 탁 곤
한국과학기술원 전자전산학과
시스템 모델링 시뮬레이션 연구실
E-mail : smcho@smslab.kaist.ac.kr, tkim@ee.kaist.ac.kr

Scheduling Feasibility Analysis Method for RT-DEVS models

Seong Myun Cho, Tag Gon Kim
System Modeling & Simulation Laboratory
Department of EECS, KAIST

실시간 시뮬레이션이란 시뮬레이션 모델의 시간 진행을 실시간에 기반하여 수행하는 시뮬레이션을 말한다. 이러한 시뮬레이션은 가상 운전 교육 프로그램 또는 컴퓨터를 이용한 컨트롤 시스템의 검증 등에 사용된다. 본 논문에서는 DEVS 형식론[Zei84]을 확장한 RT-DEVS 모델의 실시간 시뮬레이션에서 주어진 모델의 스케줄링 가능성에 대한 분석 기법을 다룬다. 제한된 시스템 리소스 상에서 여러 개의 모델을 실시간에 기반하여 시뮬레이션하려면 스케줄링이 필요하다. 실시간 스케줄링 가능성을 분석하기 위하여 시뮬레이션 모델에 제한점이 주어진다. 본 논문에서는 이러한 제한점을 알아보고 이를 만족하는 시뮬레이션 모델의 상태 궤적 그래프의 합성을 통하여 전체 시뮬레이션 시스템의 스케줄링 가능성을 알아보는 기법을 제안한다.

1. 서론

실시간 시뮬레이션이란 모델의 시간 진행을 실시간에 기반하여 수행하는 시뮬레이션을 말하며 모델은 시뮬레이션 중에 외부 프로세스와 상호작용을 갖는다. 실시간 시뮬레이션에서는 시뮬레이션 시간으로 가상 시간이 아닌 실제 시간을 사용하기 때문에 한정된 CPU 타임을 이용하여 실시간 시뮬레이션을 진행하려면 시뮬레이션 모델간의 실행 우선 순위를 정해야 하므로 실시간 스케줄링이 요구된다. 이러한 실시간 스케줄링이 가능하려면 이를 지원할 수 있는 실시간 시뮬레이션 환경이 필요하며 이전 연구에서 이러한 실시간 시뮬레이션을 지원하는 실시간 시뮬레이션 커널을 구현하였다. 본 논문에서 시스템을 모델링하기 위해 사용하는 형식론은 DEVS 형식론은

실시간 시뮬레이션에 적용되도록 확장한 RT-DEVS 형식론이다. 구현된 실시간 시뮬레이션 환경은 RT-DEVS 모델을 실시간에 기반하여 시뮬레이션 해준다. 본 논문에서는 주어진 RT-DEVS 모델의 실시간 시뮬레이션 가능성을 분석하는 방법을 제안한다. 우선 실시간 시뮬레이션 가능성 분석이 가능한 모델의 제한점을 알아보고 모델의 상태 궤적 그래프로부터 실행 가능성을 분석하는 방법을 제안한다.

본 논문은 다음과 같이 구성되어있다. 2장에서는 실시간 시스템을 모델링하기 위한 RT-DEVS 형식론에 대해서 설명한다. 3장에서는 실시간 스케줄링 분석 기법을 설명하고 4장에서 결론을 맺는다.

2. RT-DEVS 형식론

RT-DEVS 형식론은 실시간 시뮬레이션을 위하여 기존의 DEVS 형식론을 확장한 형식론이다. RT-DEVS 형식론에서 atomic 모델은 다음과 같이 정의된다.

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

, 여기서

X : 입력 사건의 집합

S : 상태 변수의 집합

Y : 출력 사건의 집합

$\delta_{ext} : Q \times X \rightarrow S$,

외부 상태 전이 함수,

여기서 $Q=\{(s,e) \mid s \in S \text{ and } 0 \le e \le ta(s)\}$

$\delta_{int} : S \times I^{-}_{0,\infty} \rightarrow S$, 내부 상태 전이 함수,

여기서 $I^{-}_{0,\infty}$ 는 음이 아닌 정수이며 모델이 어떤 상태에서 머무는 시간을 나타낸다.

$\lambda : S \times I^{-}_{0,\infty} \rightarrow Y$, 출력 함수

$ta : S \rightarrow I^{-}_{0,\infty} \times I^{-}_{0,\infty}$, 시간 진행 함수,

$\psi : S \rightarrow A$, 작업 매핑 함수

A : 작업 집합 $A = \{a \mid t(a) \in I^{-}_{0,\infty}, \text{ and } t(a) \le ta \mid_{max}\} \cup \varnothing$

기존의 DEVS 형식론에서는 시뮬레이터가 atomic 모델의 시간 진행 함수를 부를 때마다 가상의 시뮬레이션 시간이 경과하였다. 그러나 RT-DEVS 형식론에서는 가상 시간 진행 함수를 실제의 시간으로 대체한다. 즉, 시간 진행 함수가 실제 시간의 경과를 나타낸다.

RT-DEVS 형식론에서 coupled 모델은 DEVS 형식론에서의 그것과 한가지 다른 점이 있다. 그것은 DEVS 형식론에서 동시에 발생하는 사건에 대한 순서를 결정하는 SELECT 함수가 없다는 것이다. 왜냐면 실시간 시뮬레이션에서는 그러한 경우가 일어나지 않기 때문이다. 하나의 프로세서 상에서 시뮬레이션을 수행하므로 설령 외부에서 두 개 이상의 사건이 전해지더라도 오직 한번에 하나의 사건만이 처리된다.

3. 실시간 스케줄링 분석

3.1 시뮬레이션 방법론

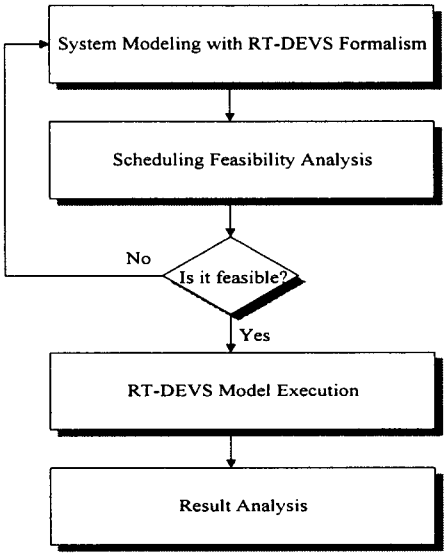

그림 1 실시간 시뮬레이션 절차

그림 1은 개괄적인 실시간 시뮬레이션 절차를 보인 것이다. 모델러는 RT-DEVS 형식론을 통해서 시스템을 구성하는 모델을 기술한다. 그런 다음 주어진 모델이 실시간 시뮬레이션이 가능한지를 알아보는 스케줄링 가능성 분석을 거친다. 이 단계에서 스케줄링 테스트 그래프가 만들어진다. 만약 주어진 시뮬레이션 모델이 실시간 시뮬레이션에 적합하지 않다고 판명되면 모델러는 실시간 스케줄링에 적합하도록 모델을 수정하는 작업을 하게 된다. 이 작업은 시간을 많이 소모하는 수행 함수를 더 작은 단위로 쪼개는 작업등을 말한다. 수정된 모델들이 실시간 스케줄링에 적합한 상태가 되면 시뮬레이션을 진행한다. 우선 스케줄러는 시뮬레이션 진행시 메시지 전달에 대한 오버헤드를 최소화하기 위하여 시스템에 포함된 모든 coupled 모델을 atomic 모델간의 연결만으로 재구성한다. 다음 각각의 모델에 해당하는 쓰레드를 생성하고 그들간의 메시지 채널을 확립한다. 실시간 제약조건을 만족된 상태에서 시뮬레이션이 마쳐지면 모델러는 얻어진 시뮬레이션 결과를 분석한다.

3.2 사건 구동 스케줄링

실시간 시뮬레이션 진행시 주어지는 시간 제약조건을 만족하기 위해서 atomic 모델의 실행 우선 순위는 최근 시간 진행 함수의 결과에 따라 결정되어야 한다. 한가지 고려할 점은 RT-DEVS 모델의 t_N(다음번 스케줄링 시간)은 실행 도중에 계속 변한다는 점이다. 따라서 모델의 실행 우선 순위는 실행도중에 동적으로 변경되어야하며 상대적으로 작은 t_N을 갖는 모델은 더 높은 실행 우선 순위를 가져야 한다. 또한 시뮬레이션 모델은 외부로부터 입력을 받을 때마다 자신의 t_N을 변경하게되므로 스케줄러는 외부 사건이 전해지는 순간 해당 모델의 실행 우선 순위를 가장 높게 정한다. 왜냐하면 모델은 외부 사건이 전해지면 자신의 t_N을 변경하고 이에 맞게 자신의 실행 우선 순위를 재조정 받아야 되기 때문이다. 따라서 내부 사건 또는 외부 사건이 모델의 실행 우선 순위 변화시키게 되므로 이러한 스케줄링 방식을 "사건 구동 스케줄링"이라 한다.

3.3 실시간 스케줄링 분석

실시간 시뮬레이션에서는 한정된 CPU 리소스로 인하여 주어진 모델이 시뮬레이션 가능하지 않을 수도 있다. 따라서 주어진 모델의 실시간 스케줄링 가능성을 알아볼 수 있는 분석 기법이 필요하다. RT-DEVS 모델의 실시간 스케줄링 분석을 위하여 다음과 같은 가정이 필요하다.

1) 시뮬레이션 도중 전체 시스템 모델의 구조는 불변한다.

2) 전체 시스템 모델은 주기적인 상태 궤적을 갖는다.

3) 컨텍스트 스위칭을 포함한 오버 헤드는 무시한다.

위에서 두 번째 가정은 RT-DEVS 모델의 스케줄링을 분석하는데 있어서 가장 기본적인 가정이다.

[정의1] 주기적인 상태 궤적

전체 상태 집합 $S = \{ s_1, s_2, \cdots, s_n \}$을 갖는 모델 $m_1, m_2, \cdots m_m$ 으로 이루어진 시스템 모델을 고려하자. 시스템의 각 상태 $s_i \in S$ 에 대하여 체류 시간(sojourn time) e_i 가 주어진다고 가정한다. 또한 전체 시스템이 s_1 s_2 s_3 ××× s_n 와 같은 상태 전이를 반복적으로 갖는다고 가정한다. 이럴 때 이 시스템은 주기적인 상태 궤적을 갖는다고 말한다.

주어진 시스템의 스케줄링 가능성을 분석하기 위하여 그림 2에 나타난 것과 같은 과정을 갖는다.

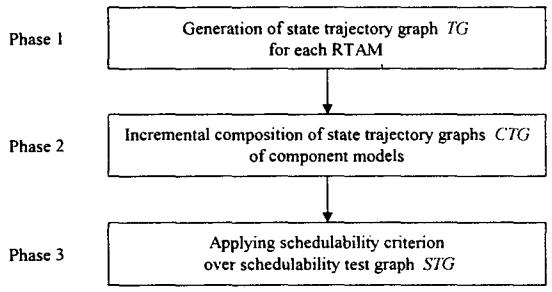

그림 2 스케줄링 가능성 분석 과정

첫 번째 단계에서 각 RTAM에 대한 상태 궤적 그래프를 얻는다. 전체 시스템의 상태 궤적은 시스템을 이루는 구성 모델의 상태 궤적에 의하여 결정되므로 전체 시스템이 주기적인 상태 궤적을 갖는다면 구성 모델들도 주기적인 상태 궤적을 갖게 된다. 이것은 RTAM에 주어지는 입력이 예측 가능하며 결정적(deterministic)이라는 것을 의미한다. 두 번째 단계에서는 이러한 구성 모델들의 상태 궤적 그래프를 순차적으로 합성한다. 시스템을 이루는 모든 구성 모델을 합성하면 전체 시스템의 상태 궤적 그래프가 얻어진다. 이 그래프가 스케줄링 검사 그래프(schedulability test graph)가 된다. 주어진 시스템의 스케줄링 가능성을 알아보기 위하여 이 그래프를 분석하게 된다.

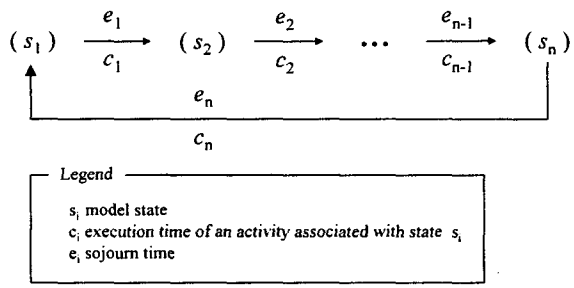

그림 3 주기적인 상태 궤적 그래프

그림 3 은 주기적인 상태 궤적을 갖는 시스템의 구성 모델의 상태 궤적 그래프를 보여준다. 여기서 c_i 는 상태 s_i 에 연관된 작업 함수의 실행 시간을 나타낸다. 시스템 구성 모델의 상태 궤적 그래프를 합성함으로써 스케줄링 검사 그래프가 얻어진다. 상태 궤적 그래프의 정의는 다음과 같다.

상태 궤적 그래프:

$TG = < S, E, C, s_0, T >$

, 여기서

S : 상태 집합

E : 체류 시간 집합

C : 실행 시간 집합

s_0 : 초기 상태

T : 전이 집합

다음과 같은 관계가 성립한다.

$e_i \in E$

$c_i = |\psi(s_i)| \in C$

$s_0 \in S$

$T \subseteq S \times E \times S$

모델이 머무는 상태는 그래프의 노드로 표시되고 상태 전이는 노드를 잇는 선으로 표시된다. 시뮬레이션 환경하에서 작업 함수의 실행시간은 $|\psi(s_i)|$ 로 표시된다.

RT-DEVS 모델은 홀로 실행되는 것이 아니므로 한 상태에서의 체류 시간은 반드시 t_{Nmax} 와 일치하는 것은 아니다. 대신 이러한 모델들은 서로 상호 작용을 갖으면서 시스템을 구성하게 된다. 전체 시스템이 주기적인 상태 궤적을 갖으면 그 구성 모델들은 시간적으로 예측 가능한 시점에서 입력을 받게 된다. 따라서 한 모델이 특정 상태에 머무는 체류 시간을 예측할 수 있다.

구성 모델들을 모두 포함한 전체 시스템의 스케줄링 가능성을 알아보려면 시스템의 상태 궤적을 얻어낼 수 있는 방법이 강구되어야 한다. 각 모델의 상태 궤적 그래프를 합성함으로써 시스템 상태 궤적 그래프를 얻을 수 있다. 상태 궤적 그래프의 합성은 다음과 같이 정의된다.

상태 궤적 그래프 합성:

$CTG = < S, E, C, T, s_0, \{TG^i\} >$

, 여기서

S : 상태 집합

E : 체류 시간 집합

C : 실행 시간 집합

s_0 : 초기 상태

T : 전이 집합

$RTAMi$: ith $RTAM$

다음과 같은 관계가 성립한다.

$S \in \times_i S^i$

$E \in \cup_i E^i$

$c_i \in C$

$s_0 = \times_i s_0^i \in S$

$T \subseteq S \times E \times S$

시스템이 머무는 상태는 그래프의 노드로 표시되고 상태 전이는 노드를 잇는 선으로 표시된다.

TG^i와 TG^j이 합성되어 얻어진 CTG의 전이 관계는 다음과 같은 규칙에 의해 정해진다.

$CTG = TG^i \parallel TG^j$

t_e를 전이 t (= $(s_1, e, s_2) \in T$) 가 발생한 시각이라하고 $t_s = t_e - e$ 로 정의하자.

규칙 1: $(s_1^i, e^i, s_2^i) \in T^i$
$(s_1^j, e^j, s_2^j) \in T^j$
where $t_s^i = t_s^j$ or $t_e^i = t_e^j$

when $e^i \le e^j$
$((s_1^i, s_1^j), e, (s_2^i, s_1^j)) \in T$
$e = \min(e^i, e^j)$

when $e^i > e^j$
$((s_1^i, s_1^j), e, (s_1^i, s_2^j)) \in T$
$e = \min(e^i, e^j)$

규칙 2: $(s_1^i, e^i, s_2^i) \in T^i$
$(s_1^j, e^j, s_2^j) \in T^j$
where $t_s^i < t_s^j$ and $t_e^i > t_e^j$

$((s_1^i, s_1^j), e, (s_1^i, s_1^j)) \in T$
$e = \min(e^i, e^j)$

규칙 3: $(s_1^i, e^i, s_2^i) \in T^i$

$(s_1^j, e^j, s_2^j) \in T^j$
where $t_s^i < t_s^j$ and $t_e^i < t_e^j$
and $t_e^i > t_s^j$

$((s_1^i, s_1^j), e, (s_2^i, s_1^j)) \in T$
$e = t_e^i - t_s^j$

합성된 모델의 전체 상태 궤적 그래프를 설명하기 위하여 그림 4에 주기적인 상태 궤적을 갖는 모델 m_1, m_2를 나타내었다. 두 모델간의 시간적인 관계를 보이기 위하여 상태 전이 다이어그램도 같이 나타내었다. 두 모델이 동시에 실행된다고 가정할 때 이 두 모델로 이루어진 시스템의 동작은 각 모델의 상태 궤적 그래프를 합성하면 알 수 있다.

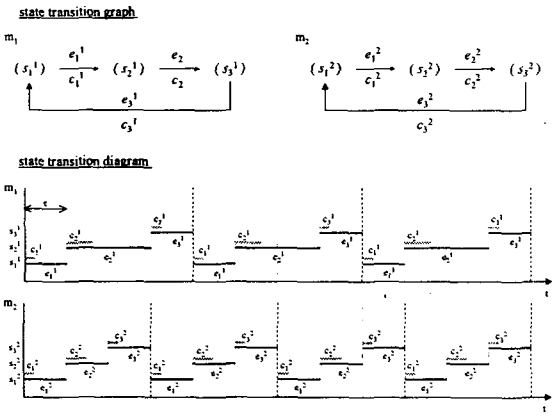

그림 4 주기적인 상태 궤적을 갖는 RT-DEVS 모델 m_1, m_2

두 모델의 전이 시각은 시간 간격 τ의 배수인 지점에서 일어난다. 합성된 그래프에서 상태 전이는 두 모델의 동작을 모두 표현하므로 스케줄링 분석에 사용될 수 있다. 이를 스케줄링 검사 그래프라고 부른다. 그림 5에 이를 나타내었다.

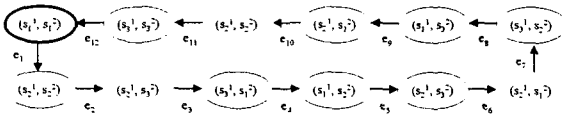

그림 5 스케줄링 검사 그래프

그림 5는 그림 4에 나타난 두 개의 모델 m_1, m_2 로 이루어진 시스템의 스케줄링 검사 그래프를 나타낸다. 구성 모델이 주기적인 상태궤적을 갖기 때문에 그것으로 합성된 스케줄링 검사 그래프도 주기적인 상태궤적을 갖는다.

스케줄링 검사 그래프에서 아래와 같이 정의한다:

[정의 2] 구분 노드
다음과 같은 조건을 만족하는 노드를 구분 노드라 한다.

1) $n_0 \in N_d$; 초기 노드 n_0 는 구분 노드이다.

2) $n_i \in N_d$, 여기서 $\forall m\ s_k^m \neq s_i^m$, $n_k \in N_d$, $n_j \notin N_d$, $k < j < i$, $1 \leq m \leq M$, M 은 모델의 개수를 나타낸다. ; 한 노드 n_i 가 이전 구분 노드와 모든 상태 변수의 값이 다를 때 이를 구분 노드라 한다.

[정의 3] 임시 노드
다음과 같은 조건을 만족하는 노드를 임시 노드라 한다.

1) $n_i \in N_t$, 여기서 $n_i \in N$, $n_i \notin N_d$; 구분 노드를 제외한 모든 노드는 임시 노드이다.

그림 5에서 원으로 표시된 노드는 구분 노드를 나타내며 그 이외의 노드는 임시 노드이다.

[정의 4] 임시 그래프 g

양 끝단의 노드만 구분 노드로 이루어진 그래프를 임시 그래프 g 라 한다. 중간의 노드는 당연히 임시 노드로 구성된다. 스케줄링 검사 그래프는 다수개의 임시 그래프로 구성된다.

스케줄링 검사 그래프는 다음과 같은 특성을 가지고 있다.

1) 노드와 노드를 잇는 선(edge)는 시간의 흐름을 나타낸다. 이것은 선에 주어지는 e_i 로 표시된다.

2) 하나의 임시 그래프의 스케줄링 가능성 분석은 다른 임시 그래프와는 독립적으로 검사할 수 있다.

3) 스케줄링 검사 그래프 G를 이루는 모든 임시 그래프가 스케줄링 가능한 것으로 판명되면 전체 시스템이 스케줄링 가능하다.

따라서 하나의 임시 그래프에 대하여 스케줄링 가능성을 분석할 수 있다면 스케줄링 검사 그래프 G를 이루는 모든 임시 그래프에 대하여 분석함으로서 주어진 시스템의 스케줄링 가능성을 알 수 있다.

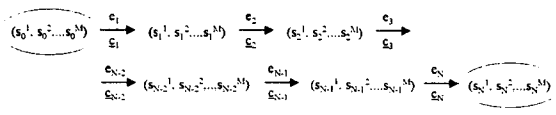

그림 6 임시 그래프의 일반적인 형태

[정리1] 임시 그래프의 스케줄링 가능성

하나의 임시 그래프 g에 대하여 $\forall i,\ e_i - c_i > 0$ 이면 g 는 스케줄링 가능하다.

여기서 e_i 노드간을 잇는 선이 나타내는 체류 시간을 나타내고 c_i 그 체류 시간동안 반드시 수행되어지는 작업 함수들의 수행 시간의 합이다.

증명:

(i) $\forall i,\ e_i - c_i \geq 0 \leftarrow$ g 는 스케줄링 가능하다.

$\exists i,\ e_i - c_i < 0$ 라고 가정하면 이것은 어떤 두 노드사이에서 작업 함수의 수행 시간이 체류 시간보다 크다는 것을 의미하게 되므로 g는 스케줄링이 불가능하다.

(ii) $\forall i,\ e_i - c_i \geq 0 \rightarrow$ g 는 스케줄링 가능하다.

g 가 스케줄링 가능하지 않다는 것은 적어도 하나의 어떤 두 노드 사이에서 작업 함수의 수행 시간이 체류 시간보다 크다는 것을 의미한다. 즉, $\exists i,\ ei - ci < 0$ 이 성립한다.

4. 결론

실시간 시뮬레이션은 시뮬레이션을 통한 운전자 교육 또는 가상의 플랜트 모델과의 연결을 통한 컨트롤러의 동작을 분석하는 작업등에 사용할 수 있다.

여러 개의 모델로 구성된 시스템을 실시간 스케줄링 하려면 한정된 CPU 리소스를 배분하기 위한 스케줄링이 필요하다. 본 논문에서는 RT-DEVS 모델의 실시간 스케줄링 분석 기법에 대하여 다루었다. 구성 모델의 상태 궤적 그래프를 기반으로 시스템을 이루는 모든 구성 모델의 상태 궤적 그래프를 합성하여 스케줄링 검사 그래프를 만들게 된다. 여기에 스케줄링이 가능성

기준을 적용하여 주어진 시스템이 실시간 시뮬레이션이 가능한지 알아보았다.

현재 제안한 실시간 스케줄링 분석 기법의 제한점은 이 분석 기법이 적용되려면 구성 요소 모델들이 주기적인 상태 궤적을 가져야 한다는 점이다. 보다 일반적인 모델로 구성된 시스템의 스케줄링 분석 기법을 개발하는 것이 앞으로의 연구과제이다.

참고 문헌

[BW94] Alan Burns and Andy Wellings, HRT-HOOD: A structured design method for hard real-time systems. *Real-Time Systems*, vol. 6, 73-114, 1994

[AV93] Alan J. Garvey and Victor R. Lesser, Design-to-time real-time scheduling. *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 5, 1491-1502, 1993

[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.

[Kim94] K. H. Kim, A utopian view of future object-oriented real-time dependable computer systems. *International Workshop on Real Time Computing Systems and Applications*, 59-69, December 1994.

[Kim97] K. H. Kim, Object structures for real-time systems and simulators. *IEEE Computer*, 30(8):62-70, August 1997.

[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*, 392-402, November 1994.

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

[LF99] Kangsun Lee and Paul A. Fishwick, OOPM/RT: A multimodeling methodology for real-time simulation. *ACM Transactions on Modeling and Computer Simulation*, vol. 9, no. 2, April, 1999, 141-170

[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. Masters 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.

[RP94] 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