特輯 技術解說

이산사건 시스템 모델링 시뮬레이션 기법

金 卓 坤
韓國科學技術院 電氣 및 電子工學科

Ⅰ. 서 언

컴퓨터 시뮬레이션은 설계자동화 도구로서 새로 제안된 설계를 증명하거나 기 설계된 시스템을 분석하여 시스템의 성능을 평가하거나 이를 최적화 하는데 효과적으로 사용될 수 있다. 컴퓨터 시뮬레이션은 실제의 시스템을 대신하는 수학적 모델을 컴퓨터를 사용하여 모의실험 함으로써 시간과 경비를 절약할 수 있을 뿐만 아니라 실제 실험에서 일어날 수도 있는 장비 및 인명에 대한 피해를 방지할 수 있는 등 여러가지 장점이 있다.

복잡한 시스템을 컴퓨터 시뮬레이션을 이용하여 설계할 경우 시스템의 설계는 여러 계층의 추상화된 레벨(abstract level)들에서 이루어지며 각 레벨에서 시스템 설계자들이 사용하는 시스템의 모델이 달라지며 시뮬레이션 목적도 또한 달라지게 된다. 예로서 VLSI 칩을 설계할 경우, 시뮬레이션은 회로 레벨, 스위치 레벨, 로직게이트 레벨, 레지스터 레벨, 그리고 이산사건 시스템(discrete event system) 레벨(혹은 간단히 시스템 레벨)등 여러 계층에서 행해진다. 이 경우 시스템 레벨을 제외한 각 레벨에서의 시뮬레이션은 SPICE(회로 레벨), VHDL(로직게이트 혹은 레지스터 레벨)등의 시뮬레이션 도구들을 사용하여 설계자가 설계된 시스템의 동작을 확인하는 것이 주 목적이다. 반면, 톱 레벨(top-level)인 시스템 레벨에서는 시스템을 component들 사이의 비동기적인(asynchronous) 상호작용에 의한 component들의 상태변이(state transition)에 초점을 맞추어 모델링하며 시스템의 동작 확인보다는 성능 평가나 이를 최적화 하는데 주로 사용된다. 따라서 이산사건 시스템 레벨에서의 시뮬레이션은 시스템의 설치시 톱 레벨 시스템 아키텍처를 결정하는데 사용되므로 아래 레벨들에서 시스템의 component들을 구체화 시키는데 중요한 역할을 한다.

본 고에서는 이산사건 시스템의 기본적인 개념과 수학적 모델링 기법 및 시뮬레이션 원리등을 다룬다. 프로그램 디버깅에 해당하는 모델 검증(verification)과 통계적 기법에 기반을 두는 모델 타당성(validation) 검토 및 입출력 테이터 분석 기법등은 취급하지 않으며 참고 문헌만을 소개한다. Ⅱ장에서는 이산사건 시스템의 정의와 수학적 모델링 기법을 간략화된 컴퓨터 시스템의 모델링을 예를 들어 기술한다. 모델링된 이산사건 시스템의 시뮬레이션 기법과 시뮬레이션 언어들이 Ⅲ장에서 고찰된다. 객체지향형 시뮬레이션 및 지식베이스를 갖는 지능형 시뮬레이션 환경이 Ⅳ장에서 기술되며, Ⅴ장의 결론에서는 본 고에서 소개되지 않았지만 시뮬레이션 모델링에 필요한 지식을 위한 참고문헌 소개가 포함된다.

Ⅱ. 이산사건 시스템의 개념 및 모델링

1. 이산사건 시스템의 정의

이산사건 시스템은 동적(dynamic) 시스템이다. 동적 시스템의 어떤 시간에서의 상태(state)는 그 시간에서 시스템이 갖는 상태변수의 값으로 결정되며, 시스템의 상태변수는 시간에 따라 변화되어 간다. 이때 시스템의 상태변수의 값이 연속적으로 변하는 시스템을 연속 시스템(continuous-change system 혹은 간단히 continuous system)이라 부르며 상태변수의 값이 어떤 특정한 점에서 순간적으로 변하는 시스템을 이산시스템(discrete-change system 혹은 간단히 discrete system)이라 부른다. 예로서, 연속 시스템 중에서 시간 영역이 연속적인 값(continuous-time)을 갖는 시스템이 아날로그 시스템이다.

이산시스템의 상태공간(state space)은 유한개의 이산적인 값으로 구성되며, 이 시스템의 시간영역은 연속적 혹은 이산적일 수 있다. 표 1은 시스템을 상태공간과 시간영역에 따라 분류한 것이다. 표 1에서 보인 바와 같이 이산사건 시스템은 시간 영역이 연속적인 실수 값을 가지며, 상태공간은 유한개의 이산적인 값으로 구성되는 이산시스템이다. 따라서, 이산사건 시스템은 임의의 시간에 순간적으로 상태변이를 일으키며 한 상태에서 머무는 시간이 불규칙적인 시스템으로 특징지을 수 있다. 따라서 운영체제 이상 레벨의 컴퓨터 시스템, 컴퓨터 네트워크, 생산시스템, C3 시스템, 그리고 교통시스템등은 모두 이산사건 시스템으로 모델링 될 수 있다.

표 1. 시스템의 분류

		Time Space	
		Continuous	Discrete
State Space	Continuous	Differential Equation System (Analog System)	Difference Equation System (DSP System)
	Discrete	Discrete Event System (Multi-Computer System)	Finite State System (Digital System)

2. 이산사건 시스템의 수학적 모델링

시스템을 수학적으로 모델링 하고자 하면 아날로그 시스템 모델링에 사용되는 미분방정식과 같은 수학적 형식론(mathematical formalism)을 사용하여야 한다. 이산사건 시스템을 모델링하기 위한 수학적 형식론에 관한 연구가 많이 있었지만, 연속 시스템에서 사용되는 미분방정식과 같은 일반적인 형식론은 존재하지 않으며, 여러 형식론들이 사용되고 있다. 이들 중 temporal 로직[26], time petri nets[29], 그리고 DEVS 형식론(discrete event systems specification formalism)[10,37]등이 이산사건 시스템을 모델링 하는데 사용되는 대표적인 형식론들이다. 본 절에서는 이들 형식론 중 이산사건 시스템을 계층적 구조를 가진 모듈화된 모델로 모델링하는 DEVS 형식론을 간략히 소개하며 보다 상세하고 formal한 의미체제(sematics)등은 참고문헌 [37]을 참조하기 바란다.

Zeigler에 의하여 제안된 DEVS 형식론은 집합이론에 기반을 두고 있으며, 이산사건 시스템을 계층적으로 분해하여 나타내기 위하여 두가지의 모델 class 즉, atomic 모델 및 coupled 모델로 나누어서 표현한다. 여기서 atomic 모델은 더 이상 분해할 수 없는 시스템 component를 표현하는 모델로 아래의 항들로 명세된다.

- 입력사건 집합(input events set)
- 출력사건 집합(output events set)
- 상태변수 집합(state set)
- 외부변이 함수(external transition function)
- 내부변이 함수(internal transition function)
- 출력 함수(output function)
- 시간전진 함수(time advance function)

입력(출력)사건 집합은 시스템의 모든 입력(출력)들의 집합을 나타내며 상태변수 집합은 시스템의 모든 가능한 상태를 포함한다. 외부변이 함수는 시스템이 외부에서 입력을 받았을 때(입력 사건이 발생할 시) 현 상태에서 다른 상태로의 상태변이를 명세하며 내부변이 함수는 시스템이 외부 입력이 없을 시 현상태에서 시간전진 함수가 정의한 시간이 경과 한 후 어떤 상태로 변이되는 가를 명세하는 함수이다. 출력 함수는 시스템이 어떤 상태에서 출력을 발생하는가를 명세하며, 시간전진 함수는 시스템이 입력사건이 발생하지 않을 시 현 상태에서 얼마동안 머문 후 다음 상태로 상태를 변이되는 가를 명세하는 함수이다.

Coupled 모델은 여러개의 component 모델 (atomic 혹은 coupled 모델)들로 구성된 복합적인 모델을 표현하며, component들의 결합방법 및 각 component 모델을 정의하기 위하여 아래의 항들로 명세된다.

- 입력사건 집합
- 출력사건 집합
- Component 각각의 모델 명세
- Coupled 모델과 component들과의 연결 명세 (coupling specification)
- Component들 사이의 연결 명세

입력 및 출력사건 집합은 atomic 모델에서와 동일하며, component 각각의 모델 명세는 각 component(atomic 혹은 coupled 모델)의 이름과 이들의 모델명세를 포함한다. Coupled 모델과 component들과의 연결 명세는 외부세계와 coupled 모델과의 통신 통로를 나타내며 component들 사이의 연결 명세는 coupled 모델 내부에서의 통신 통로를 나타낸다. 상기의 coupled 모델 명세에서 볼수 있듯이 한 coupled 모델은 다른 coupled 모델의 component가 될 수 있으므로 DEVS 형식론은 이산사건 시스템을 계층적 구조로 모델링 할 수 있다. 이상 소개한 DEVS 형식론은 시뮬레이션 언어처럼 단지 이산사건 시스템의 시뮬레이션 모델을 명세

하는 도구로만 사용되는 것이 아니라 연속 시스템에서의 미분방정식과 같이 이산사건 시스템을 수학적으로 다룰 수 있는 기법이다.

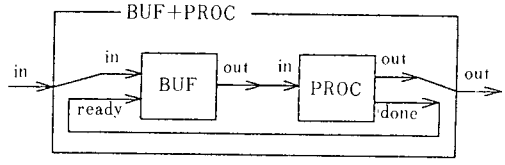
3. 예:간략화된 컴퓨터 시스템의 모델

그림 1(a)에 보인 버퍼(BUF)에 프로세서(PROC)가 종속으로 연결된 모델은 단일 프로세서를 갖는 컴퓨터 시스템의 간략화된 모델이다. 여기서 버퍼와 프로세서는 각각 atomic 모델들이고 버퍼와 프로세서를 합친 버퍼+프로세서는 coupled 모델이며 개략적인 설명은 아래와 같다. 먼저 사용자가 컴퓨터에 요구하는 계산등 제 요구사항들이 외부 입력사건 집합으로 정의되어 일련의 입력(사건)들이 임의의 시간에 버퍼+프로세서(BUF+PROC)의 입력단자 "in"을 통하여 버퍼의 입력단자 "in"에 도달한다고 하자. 이때 버퍼가 입력을 받으면 버퍼는 일단 이를 버퍼내에 있는 임시 저장 장소인 queue에 저장하고 프로세서의 현재 상태를 확인한다. 만약 프로세서가 쉬고 있으면 queue에 있는 사용자 요구사항들 중 한개를 출력단자 "out"을 통하여 프로세서에 보내고, 반면 프로세서가 현재 어떤 일을 하고 있으면 현재하고 있는 일이 끝날 때까지 기다린다. 한편 프로세서는 버퍼로부터 받은 사용자 요구사항을 일정 시간동안 처리한 후 이를 출력단자 "out"을 통하여 사용자에게 보내고 동시에 출력단자 "done"을 통하여 버퍼에게 자신이 쉬게 됨을 알려준다. 여기서 한가지 주의할 점은 버퍼는 입력사건이 발생하지 않을 시도 현재 상태에서 다른 상태로 상태변이를 한다는 점이다(DEVS 형식론에서는 이를 internal transition이라 부름). 구체적으로 말하면, 버퍼의 queue에 사용자의 요구사항들이 저장되어 있다면 이 요구사항들은 버퍼의 새로운 입력에 관계없이 프로세서가 현재 진행중인 일이 끝나는 즉시 프로세서로 보내져야 한다. 이상 설명한 버퍼의 DEVS 모델 명세는 그림 1(b)와 같다.

Ⅲ. 이산사건 시뮬레이션

1. 이산사건 시스템의 시뮬레이션 기법

이산사건 시스템의 시뮬레이션은 이산사건(입력 혹은 출력사건)이 발생하는 특정한 시간에 그 사건으로 인하여 야기되는 시스템의 상태변이를 추적하는 것이라 할 수 있다. 따라서 이산사건 시스템 시뮬레이션은 모델의 상태변이가 일어나는 시간을 미리 계획하고 계획된 시간에 상태변이를 실행시키고 아울러 다음 계획을 수립하는 등의 기능을 가진 시뮬레이션 엔진이 필요하다. 기존의 이산사건 시뮬레이션 언어들의 시뮬레이션 엔진은 크게 두가지의



(a) 간략화된 컴퓨터 시스템 모델

```
/* input port: "in", "ready"
   output port: "out"
   state variables:
     phase={SEND, PASSIVATE}
     proc_free={true, false}
     queue:FIFO                          */

/* external transition function  */
when receive an input(x, t)
   case input port of:
     "in":
        insert(x, queue)
        if length(queue)=1 and
          proc_free then
          phase:=SEND
        else continue
     "ready":
        proc_free:=true
end when ;

/* internal transition function */
case phase of:
  SEND:hold for ta(s) time unit
        queue:=rest(queue)
        waituntil proc_free
        if not empty(queue) then
          phase:=SEND
        else phase:=PASSIVATE
  PASSIVATE:waiting(x, t)

/* time advance function */
ta(s):=transmission-time(fist(queue))

/* output function */
case phase of:
  SEND:out(s)=first(queue)
  PASSIVATE:no output
```

(b) BUF의 DEVS 형식론

그림 1.

기법 즉 event-scheduling 혹은 process-interaction 접근법을 사용하고 있으며 모델링에 사용되는 world view는 이들 기법과 호환성을 가진다.

Event-scheduling 기법은 시스템내에서 사건이 일어나는 시간과 사건이 일어날 시 야기되는 모델의 상태변이를 기술하는 서브루틴(event routine이라 부름)을 한쌍으로 묶어서 시간 순으로 저장하고 (event list라고 부름) 이들 서브루틴을 시간 순서대로 실행하는 기법이다. 여기서 event list는 시뮬레이션이 진행됨에 따라 동적으로 변화되며 시뮬레이션은 event list에 event routine이 존재하는 한 계속된다.

Process-interaction 기법에서 process는 한 entity(예를 들면, Ⅱ-3절의 컴퓨터 모델에서 사용자 요구사항)가 시스템 전체를 경유하는 동안 경험하는 일련의 행위들로 정의된다. 따라서 process는 서로 연관된 일련의 사건들로 생각할 수 있으며 process내의 사건들은 두 사건 사이에 지연시간이 있으면 구별되며 그렇지 않으면 같은 사건으로 취급된다. 또한 각 process는 자신의 상태변이를 기술하는 고유의 루틴(routine)을 가지며 다른 process로부터 그 고유 루틴의 실행을 요구받으면 이를 수행하므로써 상태를 바꾼다. 따라서 process-interaction 기법에서 시뮬레이션은 시스템의 component들을 모델링하는 process들 사이의 상호 작용에 의하여 실행된다.

2. 이산사건 시스템 시뮬레이션 언어들

이산사건 시뮬레이션 언어는 위에서 기술한 시뮬레이션 엔진 뿐만 아니라 실제 시뮬레이션에 필요한 각종 난수 발생기 및 입출력 자료 해석에 필요한 통계처리 기능 등이 기본적으로 지원되어야 한다. 본 절에서는 대표적인 이산사건 시뮬레이션 언어 몇가지를 소개하고 시뮬레이션 언어 선택시 고려사항을 알아본다.

SIMSCRIPT는 1960년대에 RAND사의 Markowitz 등이 개발한 대표적인 이산사건 시뮬레이션 언어이다[7]. Fortran, Algol 및 PL/Ⅰ의 특성을 가진 이 언어는 이산사건 시뮬레이션 뿐만 아니라 Fortran처럼 일반적인 과학 계산용으로도 사용될 수 있다. 이 언어는 시스템을 entity들, 각 entity의 특성을 결정짓는 attribute들, 그리고 이들 entity들이 속하는 set들로 나타내며(이를 entities-attributes-sets world view라고 함) 시뮬레이션은 event-scheduling 기법을 사용해 왔으며 최근 process-interaction 기법을 추가하였다. 이 언어는 영어와 비슷한 문법체계를 가지며 다양한 자료 구조(data structure)와 표현력을 가지고 있다. 따라서 이 언어는 대기행렬(queueing model)로만 표현할 수 없는 방대한 이산사건 시스템을 모델링 시뮬레이션하는데 사용되며 군사 전쟁 시뮬레이션은 대부분 이 언어를 사용한 것으로 알려져 있다. 가장 최근 버전은 SIMSCRIPT Ⅱ.5이며 PC 버전이 있다.

GPSS(general purpose simulation system)는 1960대에 IBM에서 개발된 대기행렬 시뮬레이션을 위한 특수 시뮬레이션 언어이다[14]. 이 언어는 시스템을 block들과 transaction들을 사용하여 모델링하며 시뮬레이션 기법은 process-interaction을 사용한다. 이 언어는 40개 이상의 표준 블록(standard block)들의 라이브러리가 지원되며 사용자는 이들 블록을 사용하므로 모델링 절차가 간편하여 프로그래밍에 경험이 없는 사람도 비교적 쉽게 배울 수는 있지만 SIMSCRIPT나 SLAM 보다는 융통성(flexibility)이 적으며 표현력도 부족하다. 가장 최근의 IBM사 버전은 GPSS V이며 PC 버전은 GPSS/PC라 부른다[27]. 한편 1977년 Henriksen에 의해 개발된 GPSS/H는 위에 열거한 GPSS의 단점을 크게 보완했을 뿐 아니라 60개 이상의 표준 블록을 지원하며 계산 속도도 GPSS V에 비해 평균 5배 이상 빠른 것으로 보고 되어 있다[4].

SLAM(simulation language for alternative modeling)은 Fortran을 기반으로한 시뮬레이션 언어로서 1979년 Pegden과 Pritsker에 의해 개발되었다[30]. 시뮬레이션 모델링은 SLAM에서 지원하는 표준 심볼인 node들과 branch들을 사용하여 네트웍을 구성하면 된다. Node는 entity들의 생성이나 queue에 해당하며 branch는 지연시간(혹은 service time)을 나타낸다. SLAM의 시뮬레이션 기법은 event-scheduling 혹은 process-interaction 혹은 양쪽 모두를 섞어서 사용하고 있다. SLAM Ⅱ가 최근 버전이며 PC 버전은 SLAMSYSTEM이라 부른다.

SIMAN(simulation analysis)은 1982년 Pegden에 의하여 개발되었으며[28] 모델과 이를 실험하는 실험 장치에 해당하는 experimental frame을 분리하여 시뮬레이션 프로그램을 작성해야 한다는 Zeigler의 방법론[36]에 기반을 두고 있다. SIMAN의 모델은 block들로 구성되며 각 블록은 모델의 entity들 사이의 상호작용을 표현하는 로직을 기술하는데 사용된다. 시뮬레이션 기법은 SLAM과 마찬가지로 세가지 방법을 사용할 수 있으며 대부분의 경우 process-interaction 기법을 사용한다. Cinema는 SIMAN의 기능을 모두 갖추고 아울러 고도의 animation 기능을 갖춘 시뮬레이션 언어이다. 가장 최근 버전들은 SIMAN Ⅳ와 Cinema Ⅳ이며 둘 다 PC 버전이 있다.

이상의 시뮬레이션 언어들은 기능상 거의 비슷한 것들로서 대부분의 경우 한 언어로 작성된 모델은 다른 언어로 번역 가능하며 언어의 선택은 사용자의 주관적인 판단으로 정해지는 경우가 많다. 한 언어가 다른 언어들 보다 결정적으로 우수하다고 단정지을 수 없다 하더라도 모델링의 목적이나 모델의 정확도등에 따라서 언어의 선택도 달라질 수 있다.

일반적으로, 시뮬레이션 언어를 선택하는데 제일 중요한 고려사항은 주어진 언어를 사용하면 모델링을 얼마나 융통성 있게 할 수 있는가를 평가하는 것이다. 다음으로는, 주어진 언어를 사용하면 모델링을 얼마나 쉽게 할 수 있는가를 평가하는 것이다. 그밖에 고려사항들은 시뮬레이션 언어의 시뮬레이션 속도, 언어가 허용하는 최대 모델 갯수, 그리고 그 언어가 얼마나 많은 컴퓨터 기종을 지원하느냐 하는 것 등이다. 결국 한 언어가 다른 언어보다 위에서 열거한 모든 사항에서 우월할 수 없으며 모델링 시뮬레이션에 필요한 요구조건들을 적당히 trade-off 하여 언어를 선택할 수 있다. 표 2는 위에서 열거한 언어들을 비교한 것이다.

이상 열거한 시뮬레이션 언어들은 이산사건 시뮬레이션에 널리 사용되고 있다. 앞서 언급한 바와 같이, 이들 언어들은 이산사건 시스템을 표현하는 각자의 world

표 2. 이산사건 시뮬레이션 언어

Language name	Brief description	Simulation strategy	Contact	Computer(s)	O/S and other software required
GPSS	Company offers several implementations of GPSS for various equipment.	process-oriented	Simulation Software Ltd. 760 Headley Drive London, Ontario N6H 3V8 Canada Phone:(519)657-8229	VAX, MicroVAX, SUN-3, MV/ECLIPSE, ELXSI, PYRAMID	Varies
GPSS/H	Powerful, flexible, easy-to-learn language and environment; superset of well-known GPSS V.	process-oriented	Wolverine Software Corp. 7630 Little River Tumpike, Suite 208 Annandale, VA 22003 Phone:(703)750-3910	IBM-PC AT APOLLO, SUN and other work stations	MS-DOS;VM/PC; UNIX
SIMAN	General-purpose language with special features for modeling manufacturing systems.	process-oriented, event-scheduling	Systems Modelong Corp. P. O. Box 10074 State College, PA 16805 Phone:(814)238-5919	Ms-DOS compatible computers with77FORTRAN compiler	MS-DOS, OS/2, FORTRAN 77
Cinema	Simulation and animation system that enables users to create a real-time, dynamic model.	process-oriented, event-scheduling	Calder Square P.O. Box 10074 State College, PA 16805 Phone:(814)238-5919	IBM PC/AT and compatibles Micro VAX Ⅱ, SUN	MS-DOS, VMS UNIX
SLAM Ⅱ	Comprehensive simulation language permitting discrete event, continous, and network modeling.	process-oriented, event-scheduling	Michael F. Quigley Pritsker & Associates, P. O. Box 2413 West Lafayette, IN 47906 Phone:(317)463-5557	Virtually machine independent (DEC, IBM, Apollo, SUN, etc.)	FORTRAN
SIMSCRIPT Ⅱ.5	A high level, structured simulation language that can support complex models.	process-oriented, event-scheduling	Rick Crawford CACI 3344 North Torrey Pines Ct. La Jolla, CA 92037 Phone:(619)457-9681	IBM-PC, IBM mainframe, Micro VAX, VAX SUN, Prime, Sperry, CDC, Data General	None

view를 가지며 언어들의 sematics은 이들 world view를 기반으로 하고 있다. 여기서 한 가지 중요한 점은 이들 언어들의 world view는 DEVS 형식론과는 달리 수학적 형식론이 아니며 따라서 단지 이들 언어의 단지 world view를 사용한 시뮬레이션 모델은 formal 모델이 될수 없다는 것이다. DEVS 형식론을 사용한 컴퓨터 네트웍의 formal 모델을 SIMSCRIPT Ⅱ. 5를 사용하여 시뮬레이션 하는 시도가 있었으며 DEVS 형식론의 coupled 모델은 단순히 SIMSCRIPT Ⅱ. 5 혹은 위에 소개한 언어들로는 표현될 수 없음이 보고된 바 있다[8].

Ⅳ. 지능형 시뮬레이션 환경

1. 객체지향형 모델링 시뮬레이션

최근들어 객체지향형(object-oriented) 프로그래밍 기법이 소프트웨어 공학분야에서 많은 관심과 연구의 대상이 되고 있다. 객체지향형 프로그래밍 기법은 객체지향형 언어가 갖는 특징들인 data abstraction, information hiding, dynamic binding, inheritance, 그리고 polymorphism 등을 활용하여 소프트웨어 개발 시 기 개발된 소프트웨어를 "Software-IC화"하여 재 사용함으로써 소프트웨어의 생산성을 크게 향상시킨다[12].

객체지향형 프로그래밍에서는 소프트웨어 시스템을 여러개의 객체의 집합으로 생각한다. 시스템내의 각 객체는 이들이 소유하는 고유의 데이타(data)와 이에 작용되는 연산들(operations)의 한 묶음으로 정의되고 시스템은 객체들간의 메시지 전달을 명세함으로써 구현된다[12]. 이와 같은 관점은 모델링 전문가들이 이산사건 시스템을 보는 관점과 호환성을 가지며 실제로 최초의 객체지향형 언어인 SIMULA[13]은 이산사건 시뮬레이션 언어이다. 뿐만 아니라 C++를 개발한 AT & T 사의 Stroustrup도 그의 저서 [35]에서 C++는 이산사건 시뮬레이션을 목적으로 개발되었다고 밝히고 있다.

객체지향형 모델링에서는 이산사건 시스템의 실존하는 각 component들을 객체로 모델링함으로써 실제 시스템 component와 이를 모델링하는 객체가 일대일 대응 관계를 이루게 된다. 이때 각 객체는 실제 시스템 component의 특성 및 상태를 묘사하는 변수(state variable)와 이들 변수에 작용되어 component의 상태변이를 일으키는 연산들의 한 묶음으로 정의되며 이들 객체들간의 상호 작용을 명세하므로써 전체 시스템을 모델링 한다.

객체지향형 이산사건 시뮬레이션 언어(혹은 환경)는 사용자에게 위에서 열거한 객체지향형 언어가 갖는 특징들을 제공하는 이산사건 시뮬레이션 언어이다. 따라서 이들 언어의 특징을 충분히 활용하여 이산사건 시스템의 모델을 프로그래밍할 경우 모델링의 생산성을 크게 향상시킬 수 있다. 실제로 최근들어 C++를 기반으로 한 Jade사의 SIM++[15]와 Modula-2를 기반으로 한 CACI사의 MODSIM[6]등의 상용과 아리조나대학에서 개발된 DEVS-Scheme[16],[21] 등의 연구용 객체지향형 시뮬레이션 언어들이 소개되어 모델링 연구가들에게 관심을 끌고 있다.

2. DEVS-Scheme 시뮬레이션 환경

연구용으로 개발된 DEVS-Scheme은 LISP의 일종인 Scheme[1]을 기반으로 한 객체지향형 프로그래밍 시스템(SCOOPS라고 함)을 사용하여 DEVS 형식론을 구현한 객체지향형 이산사건 시뮬레이션 환경이다. DEVS-scheme은 사용자에게 DEVS 형식론을 적용하여 계층적 구조를 갖는 모듈화된 이산사건 모델(hierarchical, modular discrete event model)을 개발할 수 있는 환경을 제공한다. 이 환경은 객체지향형 기법으로 개발된 계층적 구조를 갖고 모듈화된 모델을 모델베이스 (model base)라 불리는 조직적으로 잘 정돈된 라이브러리에 보관하고 재 사용함으로써 모델링의 생산성을 크게 증진시킨다[39]. 뿐만 아니라, DEVS-scheme 환경은 사용자가 자신들의 응용에 필요한 새로운 모델 class들을 기 개발된 모델 class들을 superclass로 하여 개발 가능하게 함으로써 모델 class들을 customize할 수 있다[18],[23].

DEVS-scheme 환경의 시뮬레이션 엔진은 앞서 소개한 두 시뮬레이션 기법과는 달리 계층화된 스케쥴링(hierarchical scheduling) 기법을 사용한다. 계층화된 스케쥴링 기법은 Zeigler가 DEVS 형식론과 더불어 제안한 추상화된 시뮬레이터(abstract simulator)[37]라고 불리는 스케쥴링 알고리즘을 사용한다. 이미 설명한 바와 같이 DEVS 형식론은 두 가지의 모델 class 즉, atomic 및 coupled 모델로 시스템을 모델링하며 이들 두 모델은 서로 다른 각각의 추상화된 시뮬레이터를 사용하여 시뮬레이션 된다. 따라서 DEVS-scheme 환경내에는 두 개의 다른 추상화된 시뮬레이터 class가 존재하며 한 모델 componet 당 한개의 추상화된 시뮬레이터가 할당된다. 개략적으로 말하면, 추상화된 시뮬레이터는 모델이 언제 상태변이를 하는지 스케쥴한 다음 스케쥴된 시간에 도달하면 모델로 하여금 상태변이를 하도록 명령한다. 그림 2는 atomic 모델을 시뮬레이션하는 시뮬레이터 알고리듬이다. 여기서 "when receive(x, t)"는 시뮬레이터가 시간 t에 외부에서 입력사건 x를 받았을 시를 말하며, "when receive (*, t)"는 시뮬레이터가 시간 t에 시

간선진 함수가 명세한 스케줄된 시간이 만료되었음을 통고 받음을 의미한다. 따라서, 시뮬레이터는 (x, t)를 받았을 시 모델의 외부 변이함수를 그리고 (*, t)를 받았을 시 모델의 내부 변이함수를 각각 수행한다. Coupled 모델의 시뮬레이터도 (x, t) 및 (*, t)를 입력으로 받았을 때 이들 입력에 대하여 각기 다른 기능을 수행한다. 첫째로, 이 시뮬레이터는 외부에서 들어오는 입력(x, t)를 받았을 시 자기와 연결된 component 시뮬레이터들에게 이 (x, t)를 전달하는 기능이다. 둘째로, 스케줄된 시간이 끝났음을 알리는 (*, t)를 받았을 때 먼저 현재 어떤 component 시뮬레이터가 스케줄된 시간이 만료되는 지를 찾고, 그 시뮬레이터로 하여금 일을 끝내게 하고 다음 할 일을 계획하게 하는 기능이다. 보다 상세한 시뮬레이터 알고리즘은 [37] 을 참고하기 바란다.

그림2에 소개한 추상화된 시뮬레이터 알고리즘은 Ⅲ-2절에서 소개된 시뮬레이션 언어들이 사용하는 event list와 같은 global data structure를 갖지 않고 메시지 절달에 의하여 시뮬레이션 스케줄이 정해지는 분산 시뮬레이션 알고리즘이다. 따라서 위의 분산 알고리즘을 병렬처리형 컴퓨터 시스템 상에 구현하여 병렬 시뮬레이션 환경을 구축할 수 있으며 이에 적합한 컴퓨터 구조[11]와 시뮬레이션 성능[2]에 대한 연구 결과가 보고된 바 있다. 실제로 이들 분산 시뮬레이션 알고리즘이 Hypercube[40]와 HEP[9]등의 병렬처리형 컴퓨터에 구현된 바 있다.

```
/* t:current simulation time
   abstract simulator's state variables:
     tL:time of last event
     tN:time of next event
     e:elased time from tL
   simulated model's DEVS elements:
     s:state variable
     ext(s, e, t):external transition function
     int(s):internal transition function
     ta(s):time advance function
     out(s):output function                    */
/* ------------------------------------------------------ */
/* response to asynchronos input */
   whem receive an input(x, t)
     done:=false
       if tL<=t<=tN then
         e:=t-tN
         s:=ext(s, e, t)
         tL:=t
         tN:=tL+ta(s)
       else "error in time"
     done:=true
   end when ;
/* response to notified scheduled time */
   when recieve an input(*, t)
     done:=false
       if t=tN then
         y:=out(s)
         s:=int(s)
        tL:=t
        tN:=tL+ta(s)
       else "error in scheduled time"
     done:=true
   end whem;
```

그림 2. Atomic 모델을 시뮬레이션하는 알고리즘

3. 지식베이스된 모델링 시뮬레이션 환경

최근들어 인공지능 기술과 시뮬레이션 모델링 기법을 서로 보완적으로 사용하는 연구가 시도되고 있다. 예로서 고장 진단 전문가 시스템에서는 model based reasoning 기법을 사용한 전문가 시스템(model based expert system이라 부름)이 규칙 베이스를 사용한 전문가 시스템보다 더 정확한 진달을 할 수 있다. 역으로 규칙 베이스를 기반으로 하는 전문가 시스템은 시뮬레이션의 결과를 해석하는 등에 효과적으로 사용될 수 있다. [41]은 이러한 주제를 다루는 좋은 참고문헌이다.

지식베이스된 모델링 시뮬레이션(knowledge-based modeling simulation) 기법은 인공지능 분야에서 연구된 지식표현 기법들을 모델 표현에 적용하여 시스템을 모델링 시뮬레이션 하고저 하는 시도이다. Klahr 등[24]이 객체지향형 모델 표현에 기반을 둔 지식베이스형 시뮬레이션 시스템을 소개한 이후 많은 연구가들에 의해 유사한 시뮬레이션 시스템이 개발되었다.[17].[32].[33] 특히 [32]에서는 [24]에서와 마찬가지로 객체지향형 모델링 기법을 사용하였지만 시뮬레이션 결과를 자동 분석하는 rule base 서브시스템이 추가되어 PCB 생산 시스템등의 시뮬레이션에 효과적으로 사용되었다.

한편, [17]에서는 모델을 behavior와 structure로 분리하여 명세하는 [38]에서의 제안을 확장하여 새로운 지식표현법을 제안하고 이를 구현하여 지식베이스된 모델링 시뮬레이션 환경을 개발하였다. 제안된 지식표현법은 분

리하여 명세된 모델의 behavior와 structire를 각각 분리하여 보관하며 이들을 계층적으로 재사용(hierarchical reuse)할 수 있게 하는 기법이다. 구현된 시뮬레이션 환경은 지식베이스를 behavioral 지식베이스와 structural 지식베이스로 나누며 이들 각 지식베이스를 개발하는 도구들을 제공한다. Behavioral 지식베이스는 앞서 소개된 객체지향형 모델링 시뮬레이션 환경인 DEVS-scheme을 사용하여 개발한다. Structural 지식베이스는 시스템의 structure를 계층적으로 명세할 수 있는 지식표현법인 system entity structure 형식론[37]을 구현한 ESP-scheme [19] 환경을 사용하여 개발한다. 본 환경에서의 structural 지식베이스는 behavioral 지식베이스에 있는 모델들을 효과적으로 관리하는 일종의 모델 관리 데이타 베이스로서 복잡한 시스템의 structure를 계층적으로 명세하는데 효과적이다[22]. 구체적으로 말하면, 개발된 환경은 시스템 설계 시 여러 alternative 아키텍처를 총괄하여 하나의 계층적 structure로 명세하고 그로부터 설계 목적에 맞는 개개의 후보 모델(candidate model)들을 자동 합성하여 시뮬레이션 할 수 있게 한다. 따라서 본 환경은 여러 alternative들을 고려해야 하는 멀티 컴퓨터 시스템[20][31] 혹은 네트웍 프로토콜[34] 설계등에 효과적으로 사용될 수 있다.

V. 결 론

Man-made 동적 시스템이라 불리는 이산사건 시스템은 연속 시스템등에 비하여 우리들에게 다소 생소하며 이론적으로나 실용면에서 연구의 역사가 짧다. 특히 이산사건 시스템의 시뮬레이션은 이산사건 시스템의 이해 뿐만 아니라 특수한 시뮬레이션 언어와 이를 지원하는 컴퓨터, 그리고 긴 계산 소요시간등으로 인하여 오랫동안 일부 연구 기관등에서만 제한적으로 사용되어 왔다. 그러나, 최근들어 가격이 저렴하고 성능이 우수한 개인용 컴퓨터 시스템들의 급속한 보급과 이들 시스템에서 동작하는 시뮬레이션 언어들이 개발됨에 따라 이산사건 시스템의 모델링 시뮬레이션에 대한 관심이 급속히 확산되고 있다. 이산사건 시뮬레이션은 사건이 일어나는 순간에만 시스템의 상태변이를 추적함으로써 다른 시뮬레이션 기법에 비하여 계산이 능률적이다. 특히 서론에서 언급한 바와 같이 이산사건 모델링 시뮬레이션은 시스템을 계층적 레벨로 나누어서 설계할 시 설계 초기에 이루어져야 하며 다음 단계에서의 구체적 설계에 필요한 중요한 자료들을 제공한다. 본 고에서는 주로 이산사건 시스템의 기본 특성과 모델링 기법 및 시뮬레이션 원리등을 다루었다.

서론에서도 언급한 바와 같이 실제의 시뮬레이션 전 과정에서는 본 고에서 다룬 것들 외에도 다른 여러 지식들이 요구된다. 모델을 구현하는 컴퓨터 프로그램이 모델 명세와 똑같이 동작하는 지를 확인하는 모델 검증(verification), 모델이 실제 시스템을 얼마나 충실하게 나타내는 가를 조사하는 모델 타당성(validation) 검토, 시뮬레이션 실험을 위한 입력 자료 수집 및 분석법, 그리고 실험 결과인 출력 데이타 해석법등은 [3],[5],[25]들을 참고하기 바란다.

參 考 文 獻

[1] H. Abelson et. al., *Structure and Interpretation of Computer Programs,* MIT Press, Cambridge, MA, 1985.

[2] D. K. Baik, "Performance Evaluation and Optimal Decomposition of Hierarchical Distributed Simulation," Ph. D. Dissertation, Dept. of Computer Science, Wayne State University, Detroit, MI, 1985.

[3] O. Balci and R. G. Sargent, "A methodology for cost-risk analysis in the statistical validation of simulation models," *Comm. ACM,* vol. 24, no. 4, pp. 190-197, 1981.

[4] J. S. Banks et. al., *Getting Started with GPSS/H,* Wiverine Software Corporation, Annandale, VA, 1989.

[5] J. S. Banks and S. Carson, *Discrete-Event System Simulation,* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1984.

[6] R. F. Belanger et. al., *MODSIM Ⅱ Reference Manual,* CACI Inc-Federal, La Jolla, CA, 1989.

[7] J. E. Braun, *SIMSCRIPT Ⅱ. 5 Reference Handbook,* CACI Inc-Federal, La Jolla, CA, 1983.

[8] T. Chen, "Implementation of the Model Base Concept in SIMSCRIPT Ⅱ. 5", Master Thesis. Dept. of Electrical and Computer Engineering, University of Arizonam Tucson, AZ, 1987.

[9] A. I. Concepcion, "The Implementation of the Hierarchical Abstract Simulator on the HEP Computer," Proc. of the Winter Simulation Conference, San Francisco, pp. 428-434, Dec. 1985.

[10] A. I. Concepcion and B. P. Zeigler, "DEVS formalism: A framework for hierarchical model development," *IEEE Trans. on Software Engineering,* vol. 14, no. 2, pp. 228-241, Feb. 1988.

[11] A. I. Concepcion, "Hierarchical computer architecture for distributed simulation," *IEEE Trans. on Computers,* vol. 38, no. 2, pp. 311-319, Feb. 1989.

[12] B. J. Cox, *Object Oriented Programming: An Evolutionary Approach,* Addison-Wesley Pub. Co., Reading, MA, 1986.

[13] O. Dahl and K. Nygaard, "SIMULA-An ALGOL-based simulation language", *Comm. ACM,* vol. 9, no. 9, pp. 671-678, 1966.

[14] G. Gordon, *The Application of GPSS V to Discrete Event Simulation,* Prentice-Hall, Englewood Cliffs, N. J., 1975.

[15] Jade Simulation International Corporation, SIM++ Release 2.0 Calgary, Albert, 1989.

[16] T. G. Kim and B. P. Zeigler, "The DEVS Formalism: Hierarchical, Modular System Specification in an Object Oriented Framework," Proc. of 1987 Winter Computer Simulation Conference, pp. 559-566, Atlanta, GA, Dec. 1987.

[17] T. G. Kim, "A Knowledge-Based Environment for Hierarchical Modelling and Simulation," Ph. D. Dissertation, Dept. of Electrical and Computer Engineering, University of Arizona, Tucson, AZ, 1988.

[18] T.G. Kim and B. P. Zeigler, "The class kernel-models in DEVS-scheme: A hypercube architecture example," *ACM Simuletter,* vol. 19, no. 2, pp. 20-30, June 1988.

[19] T. G. Kim and B. P. Zeigler, "ESP-scheme: A realization of system entity structure in a LISP environment," *Advances in AI and Simulation, Simulation Series,* vol. 20, no. 4, pp. 135-140, March 1989.

[20] T. G. Kim and B. P. Zeigler, "A knowledge-based environment for investigating multicomputer architectures," *Information and Software Technology,* vol. 31, no. 10, pp. 512-520, Dec. 1989.

[21] T. G. Kim and B. P. Zeigler, "The DEVS-Scheme Simulation and Modelling Environment," Chapter 2 in *Knowledge Based Simulation: Methodology and Application* (eds: Paul A. Fishwick and Richard B. Modjeski), Springer Verlag., Inc., pp. 20-35, 1990.

[22] T. G. Kim et. al., "Entity structuring and model base management," *IEEE Trans. on Systems, Man and Cybernetics,* vol. 20, no. 5, pp. 1013-1024, 1990.

[23] T. G. Kim, "Class evolution in the DEVS-scheme simulation environment," *Expert Systems with Applications,* vol. 3, no. 3, pp. 343-351, 1991.

[24] P. Klahr and W. S. Fought, "Knowledge-Based Simulation," Proc. of First Conf. AAAI, Standford, CA, pp. 181-183, 1980.

[25] A. M. Law and W. D. Kelton, *Simulation Modeling & Analysis,* (2nd ed.) McGraw-Hill Inc., NY, 1991.

[26] J. McDermott, "A Temporal Logic for Reasoning about Processes and Plans," Cognitive Science 6, pp. 101-155, 1982.

[27] Mintutenman Software, *GPSS/PC Reference Manual,* Stow, MA, 1988.

[28] C. D. Pegden et. al., *Introduction to Simulation Using SIMAN,* Systems Modeling Corporation, Sewickley, PA, 1990.

[29] J. L. Peterson, *Petri Nets Theory and the Modeling of Systems,* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[30] A. A. B. Prisker, *Introduction to Simulation and SLAM* II, (3d ed.), Halster, NY, 1986.

[31] C. Lee et. al., "A hierarchical modelling and simulation methodology for artificial intelligence computer architectures," *Advances in AI and Simulation, Simulation Series,* vol. 22, no. 3, pp. 136-142, April 1990.

[32] Y. V. Reddy et. al., "The knowledge-based simulation system," *IEEE Software,* vol. 3, no.

2, pp. 26-37, March 1986.

[33] P. Robertson, "A rule based expert simulation environment," *Simulation Series,* vol. 17, no. 1, pp. 9-15, 1985.

[34] S. Sevinc and B. P. Zeigler, "Entity structure based design methodology: A LAN protocol example," *IEEE Trans. on Software Engineering,* vol. 14, no. 3, pp. 375-383, March 1988.

[35] B. Stroustrup, *The C++ Programming Language,* Addition-Wesley Pub. Co., Reading, MA, 1986.

[36] B.P. Zeigler, *Theory of Modelling and Simulation,* John Wiley, NY, 1976(Reissuse by Krieger Pub. Co., Malabar, FL. 1985).

[37] B. P. Zeigler, *Multifacetted Modeling and Discrete Event Simulation,* Academic Press, Orlando, FL, 1984.

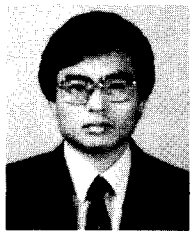
[38] B. P. Zeigler, "Knowledge representation from Minsky to Newton and beyond," *Applied Artificial Intelligence,* vol. 1, no. 1, pp. 87-107, 1987.

[39] B. P. Zeigler, *Object-oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic systems,* Academic Press, San Diego, CA, 1990.

[40] Y. Wang, "The Implem-itation of The Hierarchical Abatract Simulator on The iPSC Computer," MS Thesis, Dept. of Electrical Engineering, University of Arizona, Tucson, AZ, 1987.

[41] L. Widman, et. al.(eds), *Artificial Intelligence, Simulation & Modeling,* John Wiley & Sons Inc., New York, NY, 1989.

筆 者 紹 介



金 卓 坤

1953年 3月 3日生

1975年 부산대학교 전자공학과(공학사)

1980年 경북대학교 전자공학과(공학석사)

1988年 Arizona대학교 전기 및 전산공학과(공학박사)

1980年 9月～1983年 1月 부산수산대학교 통신공학과(현 전자공학과) 전임강사

1987年 6月～1989年 7月 미국 Environmental Research Lab. 연구원

1989年 8月～1991年 8月 Kansas대학교 전기 및 전산공학과 조교수

1991年 9月～현재 한국과학기술원 전기 및 전자공학과 조교수

주 관심 분야: 컴퓨터 시스템, 이산사건 시스템 모델링 시뮬레이션 방법론