한국시뮬레이션학회 '99추계학술대회 논문집 1999.10.9 한양대학교

DEVS 형식론을 이용한 동적 스케줄링 승강기 시뮬레이터의 구현

국민대학교 전자공학부 컴퓨터공학 실험실
권성환, 이찬수, 오하령, 성영락

Simulation of a Dynamic Scheduling Elevator System using the DEVS formalism

Kookmin University Electronic Department Computer Engineering Laboratory
Sung-hwan Kwon, Chan-su Lee, Ha-ryoung Oh, Yeong-rak Seong

요 약 본 논문에서는 다수의 승강기들이 서로 협력하여 운영되는 대형 고층 빌딩의 승강기 시스템에 대해서 DEVS 형식론을 이용하여 모델링하고 시뮬레이션한다. 다수의 독립적으로 운영되는 승강기로 이루어진 기존의 승강기 시스템과는 달리, 제안된 시스템에서는 사용자의 요구에 따라 승강기의 스케줄이 생성, 변경, 삭제되는 동적 스케줄링 기능을 갖도록 하였다. 제안된 시스템은 승강기 시스템을 나타내는 ELEVATOR 모델과 시뮬레이션 수행에 필요한 실험장치인 EF 모델로 구성된다. ELEVATOR 모델은 빌딩을 나타내는 BUILD 모델, 승강기를 나타내는 ELEV 모델과 각 승강기의 스케줄을 다이나믹하게 제어하는 SCHED 모델로 구성된다. EF 모델은 실험용 데이터를 생성하는 GENR 모델, 이동하는 승객의 데이터를 추적하고 결과를 수집하는 TRANSD 모델로 구성된다. 모델링된 시스템의 시뮬레이션을 위해 DEVSim++를 이용하여 구현하였다. DEVSim++는 DEVS 형식론을 C++로 구현한 것이다. 시뮬레이션을 통하여 제안된 시스템의 효율, 수행 시간을 기존의 승강기 시스템과 비교, 분석하였다.

1. 서 론

본 논문에서는 DEVS 형식론을 이용하여 동적 스케줄링이 가능한 승강기 시뮬레이터를 구현한다. DEVS 형식론은 이산 사건 시스템을 계층적이고 모듈화된 형태로 기술하는 수학적인 언어이다[1][2]. 본 논문의 대상 시스템은 대형 고층 건물에서 운영되는 승강기 시스템이다. 2대 이상의 승강기가 동시에 작동하는 환경을 기반으로 시스템을 작성하였으며, 다수의 승강기가 독립적으로 운영되는 기존의 승강기 시스템과는 달리, 제안된 시스템에서는 모든 승강기의 스케줄을 제어하는 스케줄러를 두었다. 이로써, 각 층 혹은 각 승강기 내부에서의 사용자 요구에 따라 각 승강기의 스케줄이 생성되고, 동작 중에도 스케줄 변경이 가능하며, 불필요한 스케줄의 경우 삭제 가능한 동적 스케줄링 기능을 갖도록 하였다.

DEVS 형식론에서는 시스템의 구성요소를 atomic 모델로 나타내고, 그들간의 연결관계 및 계층구조를 coupled 모델로 나타낸다. 본 논문에서 구현하는 승강기 시스템에서는 atomic 모델

BUILD, ELEV가 각각 빌딩, 승강기를 나타내며, 실제 승강기 탑승자를 생성시키는 GENR, 목적하는 층으로 이동하는 승객을 추적하고, 결과를 수집하는 TRANSD, 마지막으로 동적으로 스케줄을 구성, 수정하는 SCHED 등 5종의 모델을 구현하고, ELEVATOR_EF, ELEVATOR, EF 등 3개의 coupled 모델에 의해 계층적으로 전체 시스템을 구성한다.

모델링된 시스템을 시뮬레이션하기 위해 DEVS 형식론을 C++ 언어로 표현한 DEVSim++를 이용하였다[3][4]. DEVSim++를 이용하여 구현된 시뮬레이터는 스케줄을 생성한 승강기 사용자의 탑승 대기시간, 목적층까지의 이동시간, 각 승강기의 이동거리 등의 결과를 수집하여, 기존의 승강기 시스템과의 운영 효율 등을 비교하고, 검증한다.

2. DEVS 형식론

Zeigler에 의해서 제안된 DEVS(Discrete Event System Specification) 형식론은 이산사건 시스템을 기술하는 언어이다[1]. DEVS 형식론은 계층적이고 모듈화한 방법으로 이산사건 시스템의 모델들을 기술한다. 즉, DEVS 형식론은 시스템을 작은 구성요소들로 나누어 모듈화된 형태로 모델링하고 그것들을 계층적으로 구성한다. 이때 각각의 구성요소들은 atomic 모델로 표현되며 계층적인 구성은 coupled 모델로 나타낸다.

Atomic 모델은 다음과 같이 정의된다.

$AM = < X, S, Y, \delta_{ext}, \delta_{int}, \lambda, t_a >$

여기서,

X : 입력 사건의 집합, 유한집합

S : 순차적 상태변수 집합, 유한집합

Y : 출력 사건의 집합, 유한집합

δ_{ext} : 외부 전이 함수($Q \times X \rightarrow S$)

δ_{int} : 내부 전이 함수($S \rightarrow S$)

λ : 출력 함수($S \rightarrow Y$)

t_a : 시각 전진 함수($S \rightarrow R_0^{+\infty}$)

Atomic 모델의 7개의 요소 중에서 처음 3개의 요소는 시스템의 입력, 상태 그리고 출력 집합이다. 그리고 다음 4개의 요소는 앞의 3개 요소의 제약 사항을 규정한다. 또, $R_0^{+\infty}$은 0을 포함한 양의 실수의 집합이고 Q는 atomic 모델 AM의 전체상태로 다음과 같이 정의된다.

$Q = \{(s, e) \mid s \in S \text{ and } 0 \leq e \leq t_a(s)\}$

Atomic 모델은 모델의 행위를 나타내는 반면 모델의 계층적인 구조를 나타내는 coupled 모델은 복합형 구성요소로서 atomic 모델들이나 다른 coupled 모델들로 구성된다. 또한 그 자체보다 큰 규모의 coupled 모델의 구성요소가 될 수 있다. 이를 나타내기 위해 반복적으로 coupled 모델의 표현을 정의할 수 있다. 다시 말해 coupled 모델은 몇 개의 구성요소들을 결합하여 규모가 큰 모델로 확장시키는 역할을 한다. DEVS 형식론에서는 이것을 이용해서 계층적인 구조를 갖는 복잡한 모델을 쉽게 구성할 수 있게 된다. Coupled 모델의 정의는 다음과 같다.

$CM = < X, Y, M, EIC, EOC, IC, SELECT >$

여기서,

X : 입력 사건의 집합, 유한집합

Y : 출력 사건의 집합, 유한집합

M : DEVS 구성요소 모델의 집합, 유한집합

EIC : 외부입력 연결관계

($EIC \subseteq CM.IN \times M.IN$)

EOC : 외부출력 연결관계

($EOC \subseteq M.OUT \times CM.OUT$)

IC : 내부 연결관계 ($IC \subseteq M.OUT \times M.IN$)

$SELECT$: subset of $M \rightarrow M$, 여러 구성요소 모델들이 같은 시각에 스케줄을 원할 때 그 중에서 하나를 고르는 함수. 같은 시각에 내부 상태전이를 해야 할

모델이 여럿일 경우에 그것들을 순서적으로 처리하는 역할을 한다.

DEVSim++는 이산사건 시스템을 시뮬레이션하기 위해 C++언어를 사용하여 DEVS 형식론을 구현한 것으로서, 객체 지향적이고, 계층적인 시뮬레이션을 위한 환경이다. DEVSim++는 DEVS 구조하에서 계층적 합성 기술을 이용하여 이산사건 모델들을 개발할 수 있도록 지원한다. 본 논문에서는 구성한 시스템을 DEVS 형식론으로 나타낸 후 DEVSim++로 시뮬레이션 하여 적절하게 동작하는지 확인한다.

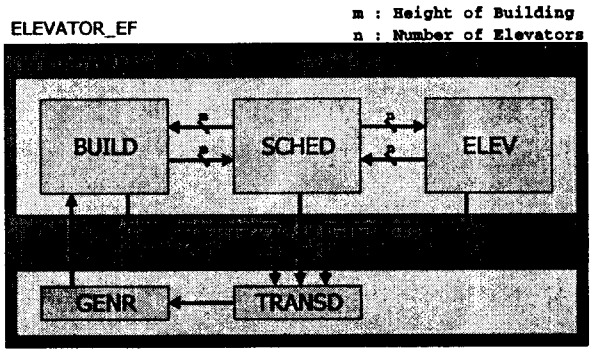
3. 모델링 및 시뮬레이터 설계

3.1 개요

기존의 승강기 시스템의 경우, 각각의 승강기들이 개별적으로 승객을 태우고, 내리는 동작을 수행하며, 이는 다른 승강기와 상관없이 개별적으로 수행한다. 이 경우, 각 승강기의 불필요한 이동이 자주 일어나게 되고, 결국 승강기 시스템 전체의 비효율을 야기 시킨다. 본 논문에서 구현한 동적 스케줄링 승강기 시스템에서는 모든 승강기의 스케줄을 제어 가능한 스케줄러를 두어, 시스템에서 작동되는 모든 승강기는 기존의 경우와 달리, 입력받은 스케줄을 스케줄러에서 저장하고, 스케줄러는 각 승강기의 스케줄을 통합 관리하게 하였다. 이로써, 사용자의 요구에 따라 각 승강기에 생성된 스케줄을 중앙 스케줄러가 통합 관리하므로, 동작 중에도 스케줄의 변경이 가능하며, 불필요한 스케줄의 경우 삭제하거나, 승강기 간 스케줄의 이동까지 가능한 동적 스케줄링 기능을 구현하였다.

3.2. 시스템 모델링

본 논문에서 구현한 승강기 시뮬레이터의 전체 시스템 구성은 아래의 그림 1과 같다.



[그림 1] 승강기 시뮬레이터의 전체 모델링

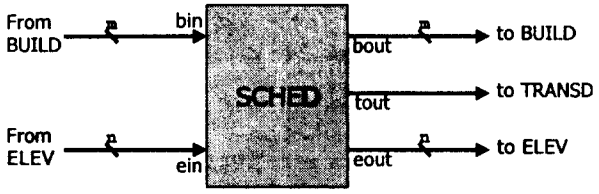
시스템은 크게 EF 모델과 ELEVATOR 모델로 구성된다. EF 모델은 시뮬레이션 과정을 위해 데이터를 생성하고, 출력된 결과를 확인할 수 있는 실험장치로, GENR, TRANSD 모델을 포함한다. GENR 모델은 실험에 필요한 데이터를 생성시키고 BUILD 모델에 전송한다. TRANSD 모델은 BUILD, SCHED, ELEV 모델의 처리 결과를 받아 정리하며, 목적했던 결과가 얻어지면, GENR 모델에 stop 신호를 보내어 시뮬레이션을 종료시킨다. ELEVATOR 모델은 빌딩, 승강기, 스케줄러를 나타내는 BUILD, ELEV, SCHED 모델로 구성된다. GENR 모델에서 탑승자와 탑승층, 목적층에 대한 정보를 받은 BUILD 모델은 새롭게 생성된 스케줄을 SCHED 모델에 통보한다. SCHED 모델은 탑승층 정보를 비교하고, 탑승층에 승강기를 위치시키면, 승강기는 탑승층에서 탑승자를 목적층까지 운송하는 과정을 통해 전체 시스템이 동작한다.

시뮬레이션 과정을 간략화하고, 시스템 효율을 객관적으로 비교할 수 있게 본 논문에서는 3대의 승강기를 사용하고, 건물은 10층으로 제한하였다. 건물의 각 층에는 up/down 버튼이 존재하여 사용자가 임의로 선택 가능하게 하였으며, 최저층

1층에는 up버튼, 최고층인 10층에는 down 버튼만을 두었다. 또한, 승강기 모델을 단순화하기 위해 승강기의 문 개폐에 관련된 open/close 버튼은 생략하였고, 이는 버튼 조작을 처리하기 위한 모델을 추가함으로써 추후에 구현 가능하도록 하였다.

3.3 SCHED 모델링

그림 1에서 나타난 바와 같이 시스템은 크게 5개의 atomic 모델과 3개의 coupled 모델로서 구성된다. 본 논문에서는 지면 관계상 가장 핵심이 되는 SCHED 모델의 예를 들어 설명하겠다.



[그림 2] Atomic 모델 SCHED

그림 2는 Atomic 모델 SCHED를 블럭도로 나타낸 것이다. SCHED는 BUILD 모델로부터 bin 포트를 통하여 사용자의 입력을 받아들이고, bout 포트로 스케쥴 테이블의 갱신 내용을 전송한다. 이를 바탕으로 스케쥴이 생성되면 eout 포트를 통하여 ELEV를 제어하고 ein 포트를 통하여 ELEV의 현재 상태를 보고 받는다. 또 시뮬레이션 결과의 분석 및 보고를 위하여 tout 포트를 통하여 TRANSD에 입출력 메시지를 복사하여 전송한다.

DEVS 형식론의 atomic 모델의 정의에 따라 SCHED를 정의하면 다음과 같다.

(1) X, Y, S

◆ X = {bin, ein}

◆ Y = {bout, eout, tout}

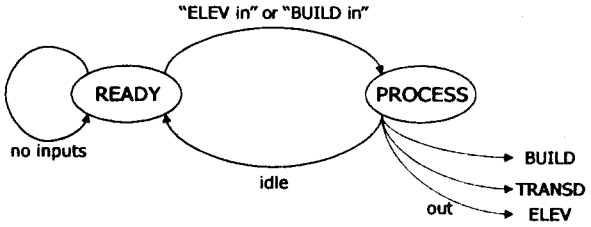
◆ S = {phase, schedTable, ...}

phase = {READY, PROCESS}

모델의 입력과 출력은 각각 그림 2의 입출력 포트에 대응한다. 그리고 SCHED는 READY, PROCESS 두가지 상태를 가지며, 기본적으로 READY 상태를 유지하고, 외부에서 입력이 들어올 경우, 이를 처리하는 PROCESS 상태로 전이되고, 처리가 끝나면 다시 READY 상태를 유지한다. 이외에 스케쥴 테이블 등을 상태 변수로서 가진다.

(2) δ_{ext} : 외부 전이 함수

외부 전이 함수는 외부로부터의 입력에 대한 각 모델의 상태 변화를 지정한다. SCHED는 ELEV 모델이나 BUILD모델에서 SCHED 모델로 입력이 들어오는 경우, SCHED 모델은 입력값을 분석하고, 승강기나 빌딩에서 추가, 삭제될 스케쥴에 대한 내용으로 스케쥴 테이블을 갱신한다.



[그림 3] SCHED의 상태 전이

(3) δ_{int} : 내부 전이 함수

내부 전이 함수는 외부 전이 함수가 발생된 뒤 변화된 상태에 해당하는 시각 전진 함수 만큼 시간이 경과했을 경우, 모델을 상태 변화를 나타낸다. SCHED 모델의 상태가 READY인 경우, 내부 전이 함수는 동작하지 않으며, 상태가 PROCESS이면 출력이후의 SCHED 모델의 상태를 READY로 전환한다.

(4) λ : 출력 함수

출력 함수는 외부 전이 함수의 결과로 변화된 상태에 해당하는 모델의 출력을 나타낸다. SCHED 모델의 경우, ELEV 모델이나 BUILD

모델에서 입력이 들어와 외부 전이 함수가 실행된 후, 각각 BUILD, ELEV, TRANSD 모델로 처리된 내용을 출력한다.

(5) t_a : 시각 전진 함수

시각 전진 함수는 모델에 외부 입력이 없을 때, 특정 상태를 유지하는 시간을 나타내는 함수이다. SCHED 모델이 READY 상태인 경우, 외부로부터 입력이 들어오지 않는 한 시각 전진 함수는 무한대를 나타내고, 상태가 PROCESS이면, PROCESSING _TIME을 리턴한다.

4. 시뮬레이션

제안된 시스템에 대한 시뮬레이션을 하기 위해서는 여러 가지의 가정이 필요하다. 임의로 지정된 값들로, 첫째, GENR 모델에서 메시지가 발생되는 주기는 1초이고, 둘째, 승강기의 한 층 이동시간 역시 1초로 가정했다. 이밖에, 승강기의 문이 여닫히는 시간과 SCHED 모델의 PROCESSING_TIME 등은 0초로 지정했다.

구현된 시뮬레이터를 테스트하기 위해 여러 방법으로 가상의 데이터를 생성하고 사용하였다. 그 중 하나의 경우를 선택하여 테스트한 시뮬레이션 결과를 나타내었다.

테스트에 사용한 데이터는 표 1과 같다. 시작층, 목적층의 값은 임의 생성되었으며, 가정한대로 메시지 발생 간격은 1초이다.

[표 1]

Customer ID	Start Floor	Dest Floor	TimeTo Call	이동거리
1	6	8	1	2
2	5	1	2	4
3	8	2	3	6
4	3	10	4	7
5	9	8	5	1
6	1	4	6	3
7	9	7	7	2
8	10	2	8	8
9	1	6	9	5
10	4	8	10	4
평균				4.2

최종 실험 결과를 정리하면 표 2와 같다.

[표 2] 실험 결과

CustomerID	TimeTo Enter	TimeTo Exit	대기시간	승차시간	합계
1	6	8	5	2	7
2	13	17	11	4	15
3	10	16	7	6	13
4	4	11	0	7	7
5	11	12	6	1	7
6	17	20	11	3	14
7	11	13	4	2	6
8	10	18	2	8	10
9	17	22	8	5	13
10	17	21	7	4	11
합계			61	42	103
평균			6.1	4.2	10.3

이상의 테스트로 승객들이 평균 이동 거리인 4.2층을 이동하는데, 대기시간 평균 6.1초, 승차시

간 평균 4.2초가 걸렸고, 전체 이동시간은 평균 10.3초가 소요되었다.

테스트 결과로 나타난 승강기의 누적 이동 거리는 다음 표 3에 나타내었으며, 각 승강기 별 누적 이동 거리값이 비슷함을 알 수 있다.

[표 3] 승강기의 누적 이동 거리

ELEV ID	Customer	Mileage
1	1, 5, 7, 8	19
2	4, 10	20
3	2, 3, 6, 9	18

5. 결론

본 논문의 목표는 동적 스케쥴링이 가능한 승강기 시스템을 모델링하고, 시뮬레이터를 작성하여, 이를 시뮬레이션 하는 것이다. 다양한 가상의 데이터로 시뮬레이션한 결과 본 논문에서 구현한 시뮬레이터는 적절히 동작하는 것을 알 수 있었다. 향후 모델의 확장에 있어, FMS, AGV의 개념을 승강기 시스템에 적용하는 방법을 통해 스케쥴관리를 보다 지능적으로 수행 할 수 있으며, 나아가 물류시스템으로의 적용도 생각해 볼 수 있다.

참고문헌

[1] Bernard P. Zeigler, Theory of Modelling and Simulation, John Wiley, 1976.

[2] Bernard P. Zeigler, Multifacetted Modelling and Discrete Event Simulation, Academic Press, 1984.

[3] 안명수, 박성봉, 김탁곤, "DEVSim++: 의미론에 기반한 이산사건 시스템의 객체지향 모델링 및 시뮬레이션 환경," 한국정보과학회논문지, 제 21 권, 제 9 호, pp. 1652-1664, 1994.

[4] Tag Gon Kim, "DEVSim++ User's Manual," KAIST, 1994.