DEVS 형식론을 이용한 물류 시스템의 성능 측정

조병헌(국민대학교), 성영락(국민대학교), 김기형(영남대학교),
오하령(국민대학교), 정성훈(한성대학교)

(Using the DEVS formalism for evaluating logistics system performance)

Byeong Heon Cho, Yeong Rak Seong, Ki Hyoung Kim,
Ha Ryoung Oh, Sung Hoon Jung

요 약

최근 기업활동의 일부로서 물류의 중요성이 급속히 부각되고 있고, 물류의 전략 개발과 모델링 방면에서 많은 연구가 진행되었다. 본 논문에서는 이산 사건 시스템을 기술하는 언어인 DEVS 형식론을 이용하여 물류 시스템의 성능을 측정하는 시뮬레이터를 구현한다. 본 논문의 대상 시스템은 다수의 차량을 이용하여 다수의 창고에서 다종의 물건들을 다수의 판매처로 운송하는 시스템이다. 각각의 창고에서는 판매처에서 요구한 물건들을 적재하여 판매처에서는 원하는 물건들을 하차하고 정해진 시간 내에 배달되는지를 검증한다. 모델링된 시스템을 시뮬레이션하기 위해 DEVSim++를 이용한다. DEVSim++는 DEVS 형식론을 C++ 언어로 표현한 것이다. 여러 가상의 데이터로 시뮬레이션한 결과 적절히 동작하는 것을 알 수 있었다. 향후 모델을 확장해서 전국적인 규모에서 시뮬레이션할 수 있는 물류 시뮬레이터를 구현하면 물류 관련 업무에서 필수적으로 쓰일 도구가 될 것이다.

1. 서론

오늘날 경제 환경의 경쟁 압력과 자원의 제한은 물류 관리를 전략적인 수준으로 올려놓았다[1]. 이는 기업활동의 일부로서 기업 물류 활동의 중요성이 급속히 부각되기 때문일 것이다. 이에 따라 최근 많은 연구들은 물류의 전략 개발 및 모델링을 목표로 하고 있고, 이 연구들은 물류의 전략, 구조 및 성능 등을 포함한다[2]. 본 논문에서는 물류 시스템의 성능을 측정하기 위해 시뮬레이터를 구현한다.

본 논문에서는 DEVS 형식론을 이용하여 물류 시뮬레이터를 구현한다. DEVS 형식론은 이산 사건 시스템을 계층적이고 모듈화된 형태로 기술하는 수학적인 언어이다[3]. 이산 사건 시스템이란 상태변화의 경우의 수는 정수 집합에 대응되고 상태변화는 무작위로 일어나는 시스템을 말하는 것으로 물류 시스템은 이 범주에 속한다. 본 논문의 대상 시스템은 다수의 차량을 이용하여 다수의 창고에서 다종의 물건들을 다수의 판매처로 운송하는 시스템이다. 각각의 창고에서는 판매처에서 요구한 물건들을 적재하여 판매처에서는 원하는 물건들을 하차하고 정해진 시간 내에 배달되는지를 검증한다.

DEVS 형식론에서는 시스템의 구성요소를 atomic 모델로 나타내고, 그들간의 연결관계 및 계층구조를 coupled 모델로 나타낸다. 본 논문에서 구현하는 물류 시스템에서는 atomic 모델로 스케쥴을 받은 차량을 보내주는 GENR, 창고를 나타내는 WARE, 판매처를 나타내는 STORE, 창고 또는 판매처에 차량이 도착하면 결과를 표출하는 TRAND, 차량을 다음 목적지로 이동시키는 DIST, 각 판매처의 주문 내용을 판매처에 보내주는 INITDIST 등 6종의 모델을 구현한다. 그리고 이

들은 EF, WARE_SET, STORE_SET, CONNECT 등의 coupled 모델에 의해 계층적으로 전체 시스템을 구성한다.

모델링된 시스템을 시뮬레이션하기 위해 DEVSim++를 이용한다. DEVSim++는 DEVS 형식론을 C++ 언어로 표현한 것이다[4][5]. DEVSim++를 이용하여 구현된 시뮬레이터는 정해진 스케쥴을 받은 차량을 메시지로 전달하여 창고에 도착하면 정해진 물건들을 정해진 양만큼 적재하고, 판매처에 도착하면 정해진 물건들을 정해진 양만큼 하차한다. 그리고 도착시간, 출발시간 등을 분석하고, 최종적으로 원하는 시간 내에 배달되었는지 검증한다.

2. DEVS 형식론

Zeigler에 의해서 제안된 DEVS(Discrete Event System Specification) 형식론은 이산사건 시스템을 기술하는 언어이다[6]. 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[5].

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

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.

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

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. DEVS 형식론에서는 coupled 모델을 이용해서 계층적인 구조를 갖는 복잡한 모델을 쉽게 구성할 수 있게 된다. Coupled 모델의 정의는 다음과 같다[5].

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

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 ($IC \subseteq M.OUT \times M.IN$)

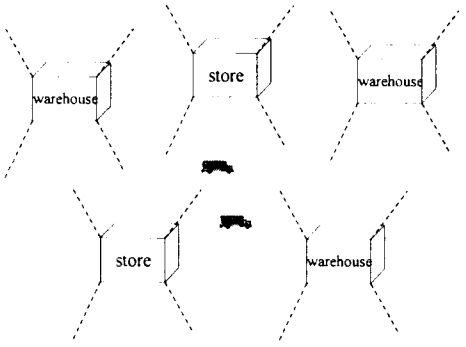
$SELECT$: subset of $M \rightarrow M$, 여러 구성요소 모델들이 같은 시각에 스케줄을 원할 때 그 중에서 하나를 고르는 함수. 같은 시각에 내부 상태전이를 해야 할 모델이 여럿일 경우에 그것들을 순서적으로 처리하는 역할을 한다.

DEVSim++[5]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 있도

록 지원한다. 본 논문에서는 구성한 시스템을 DEVS 형식론으로 나타낸 후 DEVSim++로 시뮬레이션하여 적절하게 동작하는지 확인한다.

3. 물류 시뮬레이터 설계 및 모델링

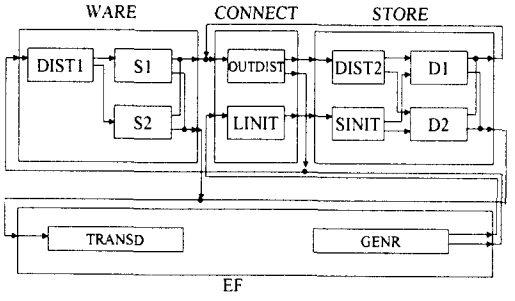
본 논문의 대상 시스템은 다수의 차량을 이용하여 다수의 창고에서 다종의 물건들을 다수의 판매처로 운송하는 시스템이다(그림1 참조). 각 차량은 들려야 할 창고와 판매처, 각 창고와 판매처에서 적재하거나 하차할 물건의 종류와 양을 스케쥴로서 갖고 있다. 물류 시뮬레이터는 정해진 스케쥴을 받은 차량을 메시지로 전달하여 창고에 도착하면 정해진 물건들을 정해진 양만큼 적재하고, 판매처에 도착하면 정해진 물건들을 정해진 양만큼 하차한다.



(그림 1) 물류 시스템

물류 시뮬레이터는 판매처에서 요구한 시간 내에 물건들이 도착하는지 검증하기 위해 창고나 판매처간의 이동 소요시간 등을 분석할 수 있도록 한다.

본 논문에서 구현하는 물류 시뮬레이터를 다음 그림과 같이 모델링하였다.



(그림 2) 물류 시뮬레이터의 전체 구성

물류 시뮬레이터는 크게 EF, WARE, CONNECT, STORE 등 4개의 coupled 모델로 구분된다. EF는 실험장치로서, 시뮬레이션을 위한 입력을 생성하고, 시뮬레이션 결과를 작성한다. EF에는 차량을 내보내는 GENR, 결과를 출력하는 TRANSD 등의 atomic 모델이 있다. 물건들을 저장하고 있으며, 차량이 도착하면 차량에 물건을 적재시키는 창고의 집합 WARE에는 창고를 나타내는 S1, S2, S3와 스케쥴에 따라 창고로 차량을 보내주는 DIST1 등의 atomic 모델이 있다. CONNECT는 WARE와 STORE를 연결시키는 역할을 한다. CONNECT에는 차량을 창고와 판매처로 구분해서 보내주는 OUTDIST, 각 판매처에 주문 내용을 보내주는 LINIT 등의 atomic 모델이 있다. 물건들을 주문하고, 차량이 도착하면 물건을 하차시키는 판매처의 집합 STORE에는 판매처를 나타내는 D1, D2, D3와 스케쥴에 따라 차량을 보내주는 DIST2, 각 판매처에 주문 내용을 보내주는 SINIT 등의 atomic 모델이 있다.

본 논문에서 구현하는 atomic 모델로서 여러 가지가 있으나 그 중 전형적인 모델로서 창고의 예를 들겠다.

창고의 atomic 모델 AM_{ware}는 다음과 같이 정의된다.

$$AM_{ware} = < X, S, Y, \delta_{ext}, \delta_{int}, \lambda, ta >$$

- ◆ X={i}
- ◆ Y={o, report}
- ◆ S={phase, queue, id, t_old=0}

  phase={busy,idle}
- ◆ δ_{ext}

```
if (ScheduleCheck() == TRUE){
  for each car in queue,
    car.tb -= t c- t_old
  t_old = tc
  tb = tn + tload
  queue.Add()
  if (phase == idle) phase = busy
}
```

◆ δ_{int}

```
if (phase == busy){
  if (queue.Length() > 0){
    for each car in queue,
      car.tb -= tc- t_old
    t_old = tc
  }
  else phase = idle
}
```

◆ λ

```
if (phase == busy){
  output report to TRANSD
  send the car with the smallest tb in queue
}
```

◆ ta

ta(busy) = the smallest t_b in queue

ta(idle) = ∞

위에서 t_c, t_n, t_{load}, t_b는 각각 현재시각, 다음 도착지까지의 운행시간, 적재시간, 다음 도착지까지의 운행시간과 적재시간의 합을 나타낸다.

외부 전이 함수는 외부로부터의 입력에 대해 모델 상태의 변화를 정의한다. 창고에 차량이 입력으로 들어오면 우선 그 차량의 스케쥴이 해당 창고에 도착할 순서인지 ScheduleCheck()라는 함수로써 검사한다. 검사 결과, 그 차량이 해당 창고에 도착할 순서라면, 물건을 적재하는 시간과 다음 도착지까지의 운행시간을 더해서 그 차량의 다음 상태 변화를 스케쥴한다. 스케쥴이 끝난 차량을 queue에 넣는다. queue내의 각 차량에는 그 케쥴 시간(t_b)을 갖고 있다. 그러므로 queue내의 각 차량의 t_b에서 경과한 시간만큼 뺀다.

내부 전이 함수는 외부 전이 함수가 발생된 뒤 변화된 상태에 해당하는 시각 전진 함수 만큼 시간이 경과했을 경우 모델 상태의 변화를 정한다. phase가 idle인 경우, 내부 전이 함수는 어떤 동작도 하지 않는다. phase가 busy이면, queue에서 t_b가 가장 작은 차량을 제거한다. 그런 다음, queue의 길이가 0이면 phase는 idle이 된다. queue의 길이가 0보다 크면 queue내의 각 차량의 t_b에서 경과한 시간만큼 뺀다.

출력 함수는 외부 전이 함수에 따라 변화된 상태에 해당하는 모델의 출력을 나타낸다. phase가 idle이면 출력 함수는 어떤 동작도 하지 않는다. phase가 busy이면 결과 분석을 하기 위해 TRANSD로 시뮬레이션 결과를 출력한다. 그리고 queue에 있는 차량 중에서 t_b가 가장 작은 차량을 출력한다.

시각 전진 함수는 모델에 외부 입력이 없을 때 어떤 상태에 얼마나 머무를 수 있는지를 나타내는 함수이다. phase가 idle이면 무한대, busy이면 queue에 있는 차량의 t_b 중 최소값을 돌려준다.

4. 시뮬레이션

물류 시뮬레이터를 테스트하기 위하여 여러 가상 데이터를 실험하였다. 그 중 하나의 데이터를 선택하여 시뮬레이션 결과를 보이겠다.

<표 1>과 <표 2>는 실험에서 사용한 각각의 판매처에서의 주문 내용과 각 차량의 스케쥴을 나타낸 것이다. 창고와 판매처는 각각 10곳, 물건의 종류는 10개로 하였다. 주문 내용에서 (x, y, t)는 물건 x를 t시간 까지 y개 주문한다는 표시이다. 예를 들어, 판매처5는 12시까지 6번 상품을 70만큼 주문하였다.

<표 1> 각 판매처에서의 주문

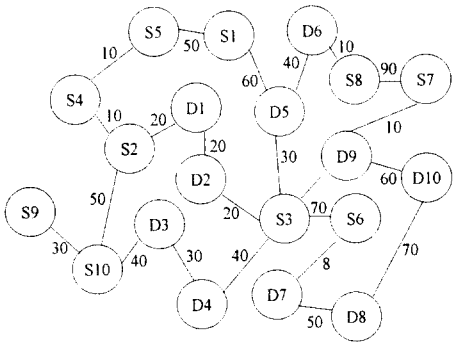
판매처	주문 내용
1	(2,20,12:00)
2	(7,20,13:00)
3	(5,10,12:00)
4	(4,100,13:00)
5	(6,70,12:00)
6	(9,60,13:00)
7	(10,50,12:00)
8	(3,50,13:00)
9	(1,20,12:00)
10	(8,90,13:00)

<표 2> 차량의 스케쥴

차량	스케쥴
1	{ (4,2,20), (2,7,20), (-1,2,-20), (-2,7,-20) }
2	{ (9,5,10), (10,4,100), (-3,5,-10), (-4,4,-100), }
3	{ (5,6,70), (1,9,60), (-5,6,-70), (-6,9,-60) }
4	{ (3,10,50), (6,3,50), (-7,10,-50), (-8,3,-50) }
5	{ (8,1,20), (7,8,90), (-9,1,-20), (-10,8,-90) }

차량 스케줄의 단위 요소 (x, y, z)에서 x가 양수이면 창고, 음수면 판매처의 ID이다. y는 물건의 종류, z는 물건의 개수이고, 양수면 적재, 음수면 하차를 뜻한다. 예를 들어, (-1,2,-20)은 판매처1에 가서 물건2 20개를 하차하라는 의미이다.

본 실험에서는 총 5대의 차량을 배정하였고, 각 창고 또는 판매처에서의 적재시간, 하차시간 등은 모두 1로 설정하였다. 그리고 각 창고 또는 판매처사이의 운송 시간은 <표 3>과 같다.



(그림 3) 각 창고 또는 판매처 사이의 운송 시간

시뮬레이션 시작 시간을 9시로 설정한 후, 위 예제로 실험한 결과, <표 3>의 결과를 얻었다.

<표 3> 실험 결과

판매처	종료시각
1	33 (9:33)
2	54 (9:54)
3	73 (10:13)
4	104 (10:44)
5	113 (10:53)
6	154 (11:34)
7	81 (10:21)
8	132 (11:12)
9	103 (10:43)
10	164 (11:44)

판매처1의 경우 <표 1>에서 물건1 1개를 12시까지 배달해 줄 것을 요구했다. 시뮬레이션 결과 9시 33분에 배달이 완료되었고, 이로써 요구시간 내에 배달되었다는 것을 알 수 있다. 마찬가지로, 나머지 판매처도 요구시간 내에 배달되었다는 것을 알 수 있다.

5. 결론

본 논문의 목표는 DEVS 형식론을 이용하여 물류 시뮬레이터를 구현하는 것이다. 여러 가상의 데이터로 시뮬레이션한 결과 본 논문에서 구현한 시뮬레이터는 적절히 동작하는 것을 알 수 있었다. 향후 모델을 확장해서 전국적인 규모에서 시뮬레이션할 수 있는 물류 시뮬레이터를 구현하면 물류 관련 업무에서 필수적으로 쓰일 도구가 될 것이다.

참고문헌

[1] Theodore P. Stank, Patrick A. Traichal, "Logistics Strategy, Organizational Design, And Performance In A Cross-Border Environment", Transpn Res.-E(Logistics and Transpn Rev.), Vol. 34, No. 1, pp. 75-86, 1998

[2] Laura Meade, Joseph Sarkis, "Strategic Analysis of Logistics and Supply Chain Management Systems Using The Analytical Network Process", Transpn Res.-E(Logistics and Transpn Rev.), Vol. 34, No. 3, pp. 201-215, 1998

[3] Tag Gon Kim, *DEVSim++ User's Manual: C++ Based Simulation with Hierarchical Modular DEVS Models*, Computer Engineering Lab., Dept. of Electrical Engineering, KAIST, 1994.

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

[5] 안명수, 박성봉, 김탁곤, "DEVSim++: 의미론에 기반한 이산사건 시스템의 객체지향 모델링 및 시뮬레이션 환경," 한국정보과학회논문지, 제21권, 제9호, pp. 1652-1664, 1994.

[6] Bernard P. Zeigler, *Multifacetted Modelling and Discrete Event Simulation*, Academic Press, Orlando, FL, 1984.