가변구조 모델링을 위한 확장된 DEVS 모델링 및 시뮬레이션 방법론

Extend DEVS Modeling and Simulation Methodology for Variable Structure Modeling

정기찬*, 이종근*, 이장세*, 지승도*

· Ki-Chan Jung, Jong-Keun Lee, Jang-Se Lee, Sung-Do Chi

Abstract

The major objective of this research is to design and build the variable structure DEVS modeling & simulation framework. To do this, we have proposed the direct message passing mechanism between the model and its simulator to deal with the structural demand from the model during the simulation. In this approach, four types of basic messages are introduced for the vertical(creation/deletion of the child) and horizontal(creation/deletion of the brother) structural changes. Proposed methodology has been successfully applied to the multi-processor system and the forest fire information system.

* 본 연구는 한국항공대학교 연구비 지원과제 "모델베이스를 이용한 지능 로보트 시스템에 관한 연구"로 수행됨.

** 한국항공대학교 컴퓨터공학과, 지능시스템 연구실

1. 서론

시뮬레이션 모델링 기법은 시스템을 좀더 현실적으로 묘사하기 위해 연구되고 사용되어 왔다. 특히 기존의 해석적 방법으로 표현하기 곤란한 각종 문제들의 동역학 연구에 있어서 컴퓨터의 발전에 힘입어 최근에 각광받고 있는 Zeigler가 제시한 DEVS 모델링 및 시뮬레이션 환경[1]은 그 발전 속도를 가속화 하고 있다. 그러나 연속적인 시간상에서 이산적으로 발생하는 사건들에 대하여 시스템의 행위를 측정하는 DEVS 모델링 및 시뮬레이션 환경은 일단 모델이 정의되어 구조가 설계되면 시뮬레이션 과정 동안 시스템의 구조적 변환이 불가능하다는 단점을 갖고 있다. 따라서 현실적으로 다양하게 변화되는 실세계를 모델링하는 연구분야에서는 시스템 설계상의 많은 제약을 안게 된다[2,3]. 그러므로 본 논문에서는 기존 시스템의 설계상 제한점들을 극복하기 위해 네 개의 가변 구조 연산자를 도입하였다. 제안된 가변구조 모델은 그 자체의 정의 내에서 구조의 변경에 관한 규칙들 즉, 첫째 시간에 따른 구조상태의 변화를 나타내는 구조적 변경과, 둘째 시간에 따른 행동상태의 변화를 나타내는 일상적 변경을 포함함으로써 시뮬레이션 수행 중에 행동적 변화뿐 아니라 구성원과 그들 간의 결합관계에도 변화를 나타낼 수 있을 뿐 아니라[2,3], 모델들이 생성, 삭제, 변화의 진화적인 현실성을 나타낼 수 있다[4]. 한편 이러한 가변구조에 대한 연구로는 지능적 구조화 처리기를 이용하여 필요한 실체 구조의 가지치기를 하여 새롭게 입력되는 목적들에 반응할 수 있는 다쪼각 개념을 이용한 가변구조 모델링[2]과 다계급을 가지는 시스템처럼 상위 계층이 하위 계층의 구조를 통제할 수 있도록 시뮬레이션 개념을 정립하는 계층구조적 다단계 시스템 이론[3], 또한 모델을 네트워크로 표현하며 실행 모델을 통하여 가변구조에 접근한 DSDEVS[5,6]가 있으며, 모델의 동질성 관계를 통한 가변구조 모델로의 접근[7]이 시도된 바 있다. 이와같이 모델구조의 변화를 수행시킬 수 방법으로는, 첫째 DSDEVS와 같이 하나의 모델이 다른 모델들을 제어(자식/형제의 생성,삭제)하는 중앙집중방식, 둘째 계층 구조적 다단계 시스템과 같이 상위 계층이 하위 계층을 제어하는 방식, 셋째 본 논문에서 제안하는 각각의 모델들이 내부 연산에 의한 요구사항에 따라 자신의 형제와 자식을 생성하는 분산방식 등을 들 수 있다. 한편 기존의 방법들은 지능적 구조화 처리기나 상위 계층이 모델들의 상태를 모니터링하여 구조를 변경하는 방식을 사용함으로 일괄적인 처리가 가능한 장점을 갖는 반면 가변구조 요구가 증가하고 하위 계층이 복잡해질수록 제어하기가 어렵게 되는 단점을 가진다. 그러나 각각의 모델이 내부 연산에 의한 자체적인 변화를 하여 모델을 생성하는 분산방식에서는 구조변경에 덜 민감하기 때문에 시스템의 확장성에 많은 장점을 가지게 된다. 본 논문은 기존의 DEVS 모델링 및 시뮬레이션 환경이 갖는 고정된 구조로 인한 비효율성들을 새로운 가변구조 연산자를 도입함으로써 동적인 변화에 보다 효과적으로 대처할 수 있는 확장된 DEVS 모델링 및 시뮬레이션 방법론을 제시하고 이를 통하여 구현된 D^2-SIM을 소개한다.

본 논문의 순서는 먼저 2장에서 가변 구조 모델링에 대한 방법론에 대한 소개를 하고, 3장에서 가변 구조 D^2-SIM를 구현하기 위한 방법론에 대해서 논하고, 4장에서는 사례연구를 통해 제안된 방법론의 적용 가능성을 검증하고 결론을 맺는다. 마지막으로 부록에서는 가변구조를 구현한 DELTA 시스템과 D^2-SIM의 비교분석을 수행하였다.

2. 가변구조 모델링 방법론

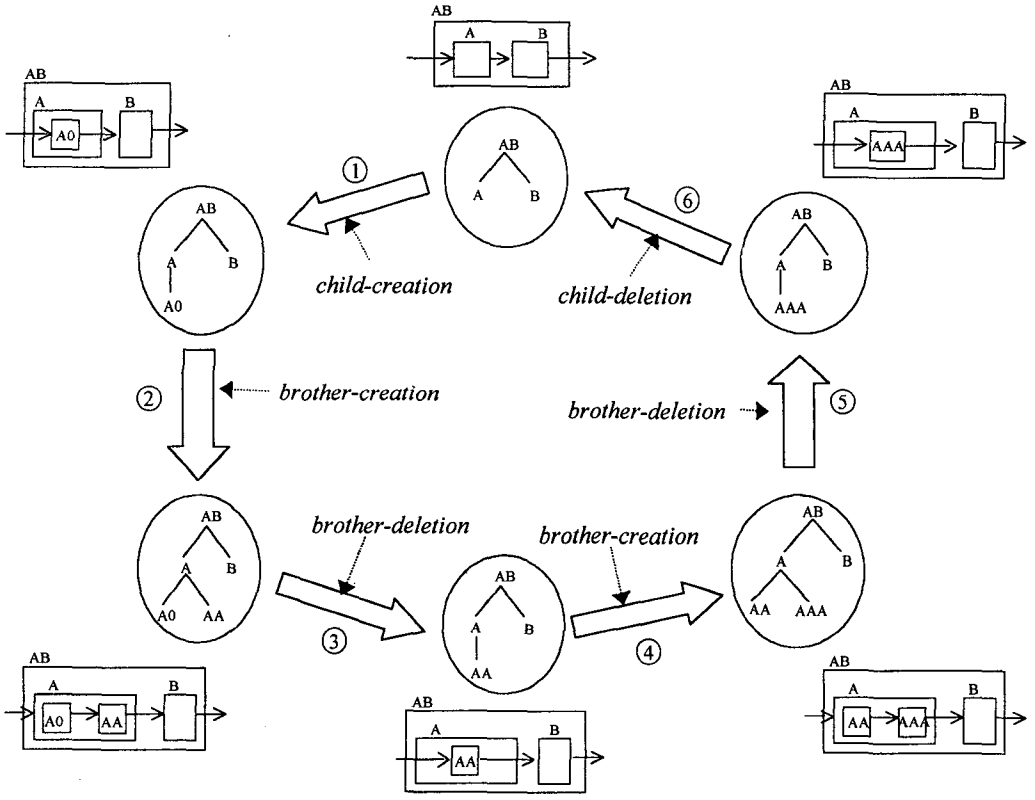
이산 사건 시스템 모델링을 위한 대표적인 형식론인 DEVS모델은 연속적인 시간상에서 이산적으로 발생하는 사건들에 대하여 시스템의 행위를 측정하는 것으로서 다음과 같은 형식론에 의해 모델을 표현한다. 즉, 입력집합 X, 출력집합 Y, 상태집합 S, 시간진행함수 ta, 외부상태전이함수 δ_{ext}, 내부상태전이함수 δ_{int}, 출력함수 λ 등의 7개의 구성요소로 이루어진다[1,8,9].

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

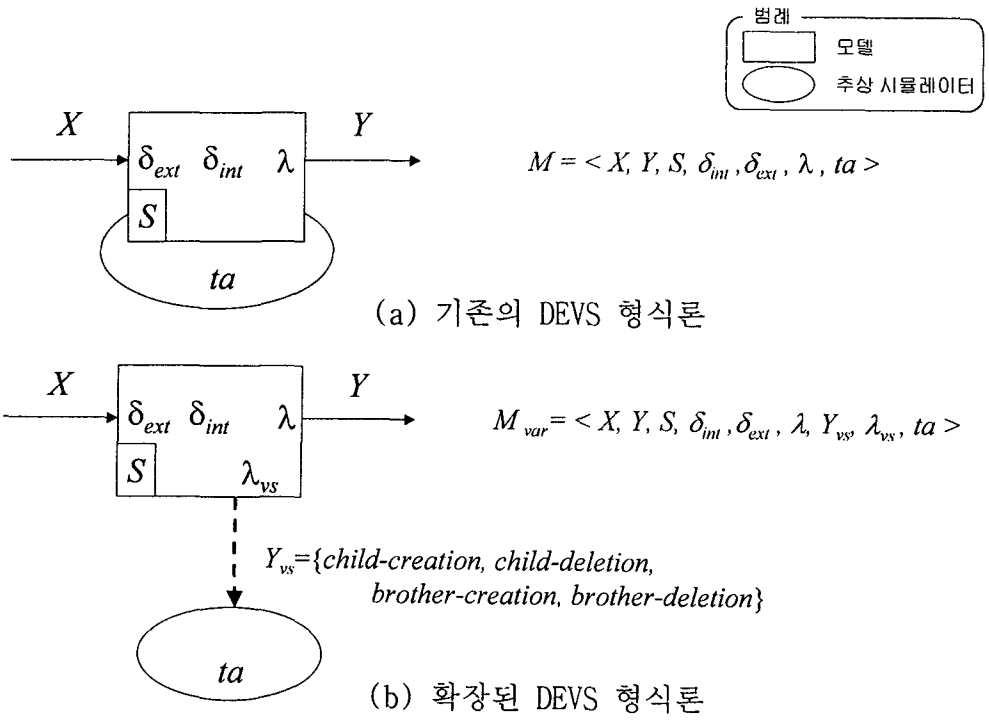
가변구조 모델은 이러한 구성요소 외에 자체의 정의 내에서 구조의 변경에 관한 구성요소(가변구조출력함수 λ_{vs}, 가변구조출력집합 Y_{vs}를, 여기서 Y_{vs}={*child-creation, child-deletion, brother-creation, brother-deletion*}, 포함함으로써 시간에 따라 변화하는 환경에서 자신의 형태를 환경에 적합하게 변화시킬 수 있다. 또한 시뮬레이션 수행 중에 행동적 변화뿐 아니라 구성원과 그들 간의 결합관계에도 변화를 나타낼 수 있으며, 외부의 결정에 영향을 받지 않고 모델 내부 연산에 의한 자체적인 변화가 가능해진다. 따라서, 확장된 가변구조 DEVS 모델은 다음과 같이 정의된다.

$$M_{var} = < X, S, Y, \delta_{int}, \delta_{ext}, \lambda, Y_{vs}, \lambda_{vs}, ta >$$

확장된 가변구조 DEVS 모델의 가변구조출력집합(Y_{vs})에 속한 4개의 가변구조 메시지를 통해서 시스템의 구조가 변화되는 예가 <그림 1>에 있다. A와 B모델 두 개를 가지고 있는 최초의 모델 AB로부터 시작하여 원자모델(atomic-model)인 A모델 자신이 *child-creation-message*를 통하여 자식 모델을 하나 생성시킬 것을 요구하면 자신은 연자모델(digraph-model)인 부모모델로 변환되고 A 모델 안에 A0모델이 생성 되게 된다(화살표①). 다음으로 A0모델이 *brother-creation-message*를 통하여 자신의 형제 모델을 생성시킬 것을 요구하면 A0의 형제인 AA모델이 생성되며(화살표②), 또, AA모델이 자신의 형제를 삭제시킬 것을 요구하는 *brother-deletion-message*를 통하여 A0모델을 삭제한다(화살표③). 마찬가지로 AA모델은 자신의 형제 모델을 생성시킬 것을 요구하는 *brother-creation-message*를 통하여 AA의 형제인 AAA모델을 생성하고 (화살표④) *brother-deletion-message*에 의하여 AAA모델의 형제를 삭제시킬 것을 요구하면 A



<그림 1> 가변구조 모델링의 예

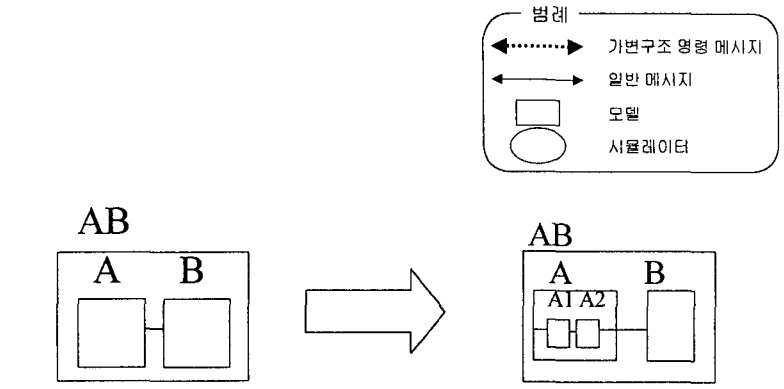


<그림 2> 기존의 DEVS 형식론과 확장된 DEVS 형식론 비교

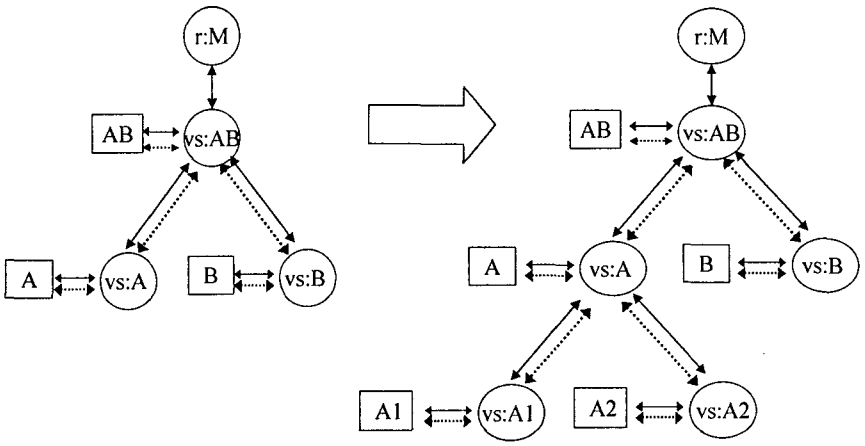
A모델이 삭제가 된다(화살표⑤). 끝으로 연자모델인 부모 모델(A모델)이 자신의 자식모델(AAA모델)의 삭제를 요구하고 자식이 없으면 원자모델로 변환되는 *child-deletion-message*를 통하여 처음 시작했던 A, B 두 개의 모델을 가진 최초의 모델 AB가 된다(화살표⑥). 이와 같이 4개의 구조변환 메시지를 이용함으로써 자식과 형제에 대한 생성과 삭제가 가능한 가변구조가 달성될 수 있다.

<그림 2>는 확장된 DEVS 형식론과 기존의 DEVS 형식론의 비교를 나타낸다. 확장된 DEVS 형식론은 모델의 가변구조출력함수(λ_{vs})와 가변구조 출력집합(Y_{vs})을 통해서 추상 시뮬레이터[8]에게 가변구조 메시지를 전달함으로써 구조를 변화할 수 있는 구조변환 연산을 수행할 수 있기 때문에 가변구조가 가능하게 된다.

<그림 3>은 모델이 새롭게 생성되기 전과 생성된 후에 가변구조의 시뮬레이션 모델 구조를 나타낸 것으로 가변구조 명령 메시지에 의하여 모델 A1과 A2가 생성되며 각각의 모델에 가변구조 시뮬레이터가 생성되는 것을 보인다. 가변구조 시뮬레이터의 알고리즘은 그림 4에 나타나는데, 알고리즘에서 전달이 되는 메시지에는 기본적으로 4개가 있다. 첫 번째 *x-message*는 외부 사건의 도착을 표현하고 두 번째 *done-message*는 부모에게 상태의 변화가 발생한 것을 알려주며, 세 번째 *y-message*는 내부상태변환이 일어나기 전에 발생하는 출력을 표현하고 마지막 **-message*는 변화해야할 다음 내부 이벤트 사건에 대해서 표현한다[1]. 여기에 가변구조를 위해 4개의 message를 추가하였는데, 첫 번째 *child-creation-message*는 자식의 생성요구를 표현하고 두 번째 *child-deletion-message*는 자식의 삭제요구를 표현하며, 세 번째 *brother-creation-message*는 부모에게 형제생성에 대한 요구를 표현하고 마지막 *brother-deletion-message*는 부모에게 형제삭제에 대한 요구를 표현한다.



(a) 모델 구조 변환



(b) 시뮬레이터 구조 변환

<그림 3> 시뮬레이션 구조 변환의 예

그림에서 4개의 가변구조 메시지인 *brother -creation, brother-deletion, child-creation* 그리고 *child-deletion-message*는 **-message* 처리함수에서 가변구조 출력 함수를 통해서 반환 받게 되는데 그 중에서 *child-creation-message*에 대한 처리는 **-message* 처리 함수에서 그리고 나머지 message에 대한 처리는 *x-message* 처리함수에서 하게 된다. (점선 표시)

```
When receive an input(*, t)
 done := false
    if tag := vs_atomic then
    if t := tN then
    y :=   λ(s)
    yvs := λvs(s)
    if yvs := child-creation
              tag := vs_coupled
              create model
              model coupling
    if yvs := child-deletion
              yvs := #cd
    else if yvs := brother-creaion or yvs := brother-deletion
              yvs := #bc or yvs := #bd
    s := int(s)
    tL := t
    tN := tL + ta(s)
    else error
Else if tag := vs_coupled then
    if t:= tN then
    Find the simulator with minimum tN
    SELECT one, I* and send the input(*, t) to it
    Send the signals(xi*, j, t) to each of its influences
    wait until this simulator I* and each of its influences done
    tL := t
    tN := minimum of component tNs
    done := true
 end when
```

(a) **-message* 처리 알고리즘

```
When receive an input(x,t)
done := false
    if tag := vs_atomic then
    if tL < t < tN then
      e := t - tL
      s := ext(s,e,x)
      tL := t
      tN := tL + ta(s)
    else error
else if tag := vs_coupled then
    if tL < t < tN then
    send input(x, t) to each component simulator I
    wait until all simulators done
    if yvs := #bc then
        create model
        model coupling
    else if  yvs := #bd then
         delete model
         model uncoupling
    else if yvs := #cd
         tag := vs_atomic
         delete model
         model uncoupling
    tL := t
    tN := minimum of component tNs
    else error
done := true
end when
```

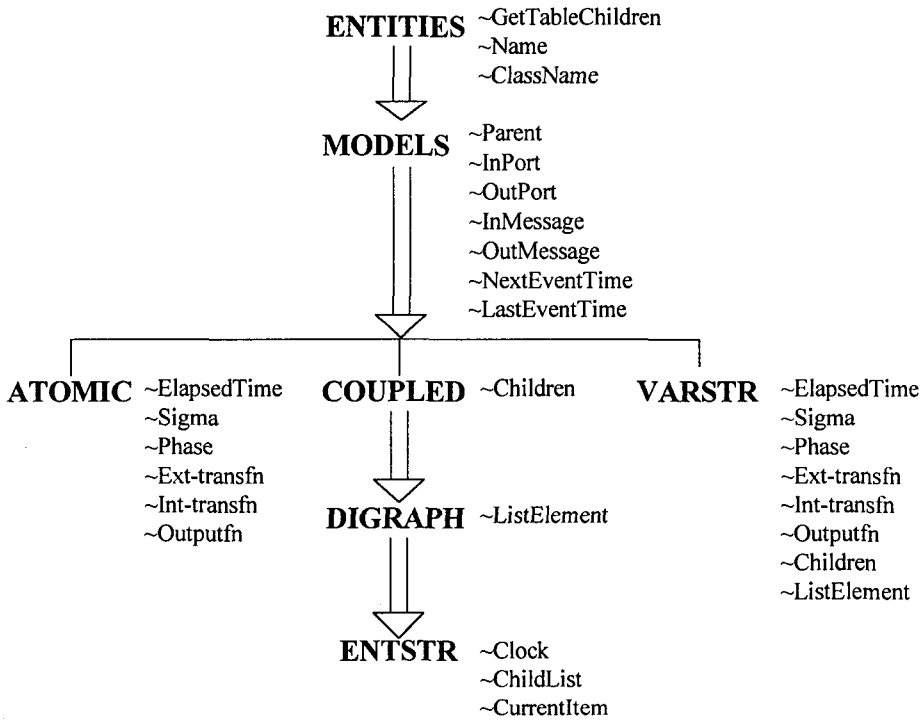
(b) *x-message* 처리 알고리즘

<그림 4> 가변구조 시뮬레이터 알고리즘

3. D^2-SIM 구현 방법론

가변구조 DEVS 모델링 및 시뮬레이션 프레임워크는 기존의 DEVS 프레임워크의 구현체 중의 하나인 D-SIM[11]에 가변구조에 대한 처리를 할 수 있는 새로운 클래스인 VARSTR를 추가함으로 구현하였다. 다음은 D^2-SIM(Dynamic Devs SIMulation) 클래스에 대한 설명을 하고 있다. 또한 가변구조를 위한 확장된 D^2-SIM의 클래스 구조는 <그림 5>와 같다.

ENTITIES : 모든 DEVS 요소의 추상적인 클래스로서 요소들의 이름과 클래스 이름만을 갖는다.
MODELS : 모든 DEVS 모델의 추상적인 클래스로서 부모 모델의 포인터, 결합정보, 시뮬레이션 시간을 관리한다.
ATOMIC : 모든 실제적인 응용모델의 추상적인 클래스로서 모델의 상태와 시뮬레이션 시간을 관리하며 네 개의 가상 함수를 갖는다. 모델 설계자는 반드시 이 클래스의 상속에 의해서 모델 클래스를 만들어야 한다.
COUPLED : 자식 모델을 가지고 있는 모든 모델의 추상적인 모델로써 자식을 관리한다.
DIGRAPH : 자식모델의 명단을 멤버 변수로 가지고 있는 클래스로서 모델 설계자는 반드시 이 클래스의 하나 이상의 요소를 사용해서 자식모델과 결합을 해야한다.
ENTSTR : 전반적인 시뮬레이션을 관리하는 클래스로서 모델 설계자는 시뮬레이션을 위해서 이 클래스 중의 몇 개를 요소로 가져야 한다.
VARSTR : ATOMIC과 COUPLED의 기능을 동시에 갖는 클래스로서 모든 응용모델과 자식을 가지고 있는 모델의 추상적인 클래스이다. ATOMIC과 COUPLED 공통의 함수에서 각자의 역할변환은 tag값을 가지고 구별하게 된다. 모델 설계자는 이 클래스에서 상속을 받아서 가변구조가 가능한 모델



<그림 5> D^2-SIM 클래스 구조

을 만들 수 있으며, 이것은 네 개의 가상함수를 가지고 있다(ExtTransitionFN, IntTransitionFN, OutputFN, and InitializeFN).

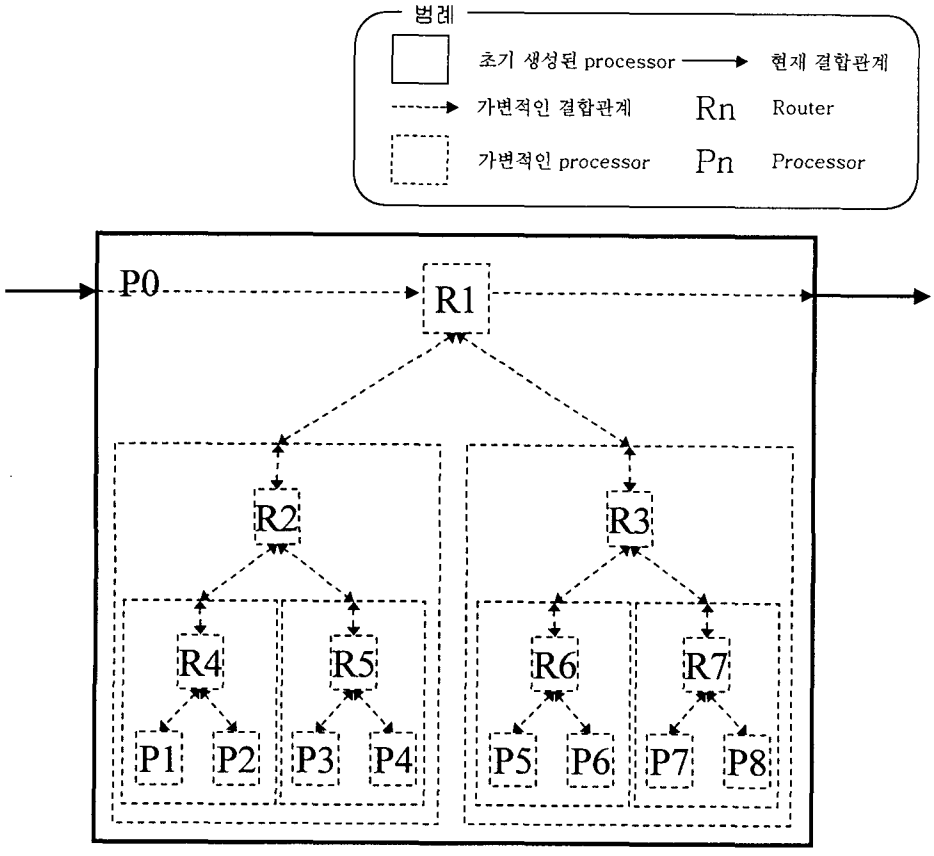
4. 사례연구

본 장에서는 제안된 가변구조 방법론을 이용한 다중 프로세서 시스템과 산불정보시스템에 대한 적용 사례를 설명한다.

4.1 가변구조형 다중 프로세서 시스템

가변구조형 다중 프로세서 시스템은 초기에 하나의 프로세서로부터 시작하여 Job의 입력상황에 따라 가변적으로 자신의 프로세서의 개수를 변화시켜 시스템의 성능저하 없이 Job 처리능력을 유지하려는 시스템으로 <그림 6>과 같은 형태를 가지면서 계속적으로 확장될 수 있다. 가변구조형 다중 프로세서 시스템에 대한 시뮬레이션 초기 조건은 <표 1>과 같다.

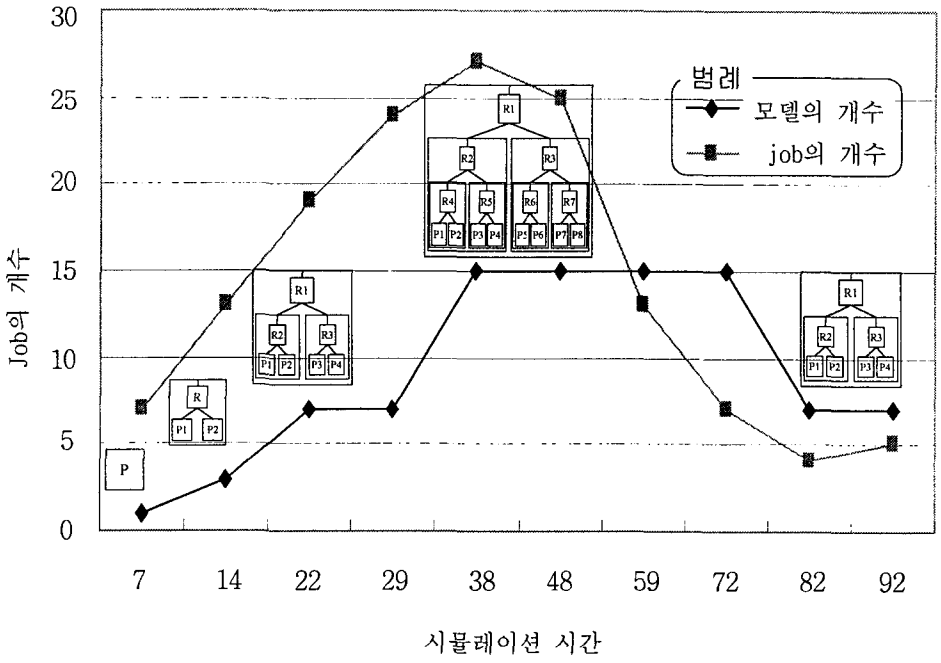
이러한 조건을 통하여 얻은 시뮬레이션 결과는 <그림 7>과 같다. 초기에는 하나의 프로세서로 Job을 처리하는데, 프로세서에 있는 큐의 Job이 5개 이상이 되면 첫 번째 단계로 프로세서 안에 자식생성(*child-creation*)과 형제생성(*brother-creation*)을 하여 구조를 변화한 후 router와 두 개의 프로세서로 시뮬레이션을 수행한다(time=14). 두 번째 단계로 두 개의 프로세서에 있는 큐의 Job이 각각



<그림 6> 가변구조형 다중 프로세서 시스템 구조의 예

<표 1> 가변구조형 다중 프로세서 시스템을 위한 시뮬레이션 초기 조건

변수	조건	비고
Job 평균 발생시간(sec)	1.0	Job이 35개 이상이면 발생간격 5.0으로 증가
Processing Time(sec)	7.0	평균 Job처리 시간
Active Time(sec)	100	실험시간
구조 변형 가능 큐의 Job 개수(N)	N >= 5	5개 이상/이하면 busy/passive로 간주
Router 한 개당 프로세서	2개	



<그림 7> 가변구조형 다중 프로세서 시스템의 구조 변화 예

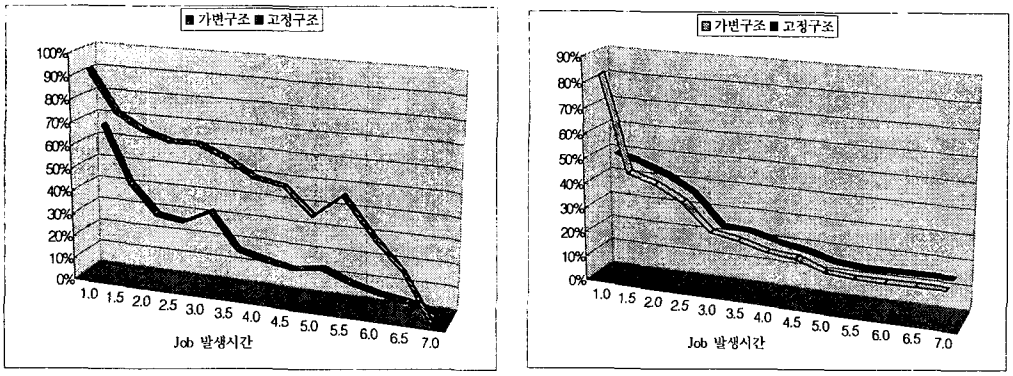
5개 이상이 되면 각각의 프로세서 안에 자식생성과 형제생성을 하여 구조를 변화한 후 4개의 프로세서로 Job을 처리한다(time=22). 세 번째 단계로 이 4개의 프로세서 각각의 큐가 Job을 5개 이상으로 가지면 다시 프로세서 안에 자식생성과 형제생성을 하여 구조를 변화한 후 총 8개의 프로세서로 Job을 처리한다(time=38). 네 번째 단계로 Job 평균 발생시간이 1.0에서 5.0으로 변화하는 35번째 Job 이후에 큐의 Job 개수가 적어지게 되면서 개수가 0이 되면 router가 형제삭제(*brother-deletion*)를 하고 자신도 삭제(*child-deletion*) 되어서 다시 4개의 프로세서가 일을 하는 형태로 구조가 변하게 된다(time=82).

가변구조형 다중 프로세서 시스템과 고정구조형 다중 프로세서 시스템에 대하여 <표 2>에서 제시한 시뮬레이션 조건을 통하여 비교분석 하였다.

그림 8(a)의 결과에서 보듯이 고정구조 방식의 경우 사용되지 않는 프로세서가 존재하기 때문에

<표 2> 큐 활용도와 Job 처리율 비교분석을 위한 시뮬레이션 초기 조건

변수	조건	비고
Job 평균 발생시간(sec)	1.0 - 7.0	0.5씩 증가
Processing Time(sec)	6.0	평균 Job처리 시간
Active Time(sec)	100	실험시간
구조 변형 가능 큐의 Job 개수(N)	N >= 5	5개 이상/이하면 busy/passive로 간주



(a) 큐 활용도의 비교 결과 (b) Job 처리율의 비교 결과

<그림 8> 가변구조와 고정구조 시뮬레이션 결과

필요시에만 모델이 생성되는 가변구조 방식에 비해서 전체적인 큐 활용도가 떨어진다. 그림 8(b)에 나타난 바와 같이 Job 처리율의 비교 결과를 보면 Job 평균 발생시간 2.0에서 7.0까지는 가변구조 방식과 고정구조 방식의 Job 처리율이 유사하지만 Job 평균 발생시간이 2.0보다 적은 경우, 즉 1.0에서 1.5사이에서 고정구조 방식은 더 이상의 사용할 수 있는 모델이 없기 때문에 Job을 처리하지 못하고 그냥 큐에 두다가 결국은 한계치에 이르러 Job을 잃어버릴 수 밖에 없어진다. 한편 가변구조 방식은 새로운 모델을 생성해서 계속적으로 균등하게 모델들이 일을 처리할 수 있기 때문에 2.0보다 적은 Job 평균 발생시간에서는 가변구조가 고정구조에 비해서 Job 처리율이 높아지게 된다.

이와 같이, 가변구조형 다중 프로세서 시스템은 필요에 따라서 자신의 구조를 변화시킴으로써 동적인 변화에 보다 효과적으로 대처할 수 있다. 또한 다중 프로세서를 가변구조와 고정구조에 대해서 동일하게 시뮬레이션 수행을 하였을 때 큐 활용도와 Job 처리율에 대하여 성능분석을 한 결과 정해진 프로세서의 개수 이상의 요구가 들어왔을 때 고정구조에 비해서 가변구조가 모든 면에서 효율적이며 유연성 있게 대처한다는 것을 확인 할 수가 있었다.

4.2 산불 정보 시스템

산불 정보 시스템[12,13]은 산불 발생시 각종 센서들로부터 입수되는 데이터(발화지점, 범위 등)들과 산림청, 기상청, 그리고 관할 소방본부 등 관련 기

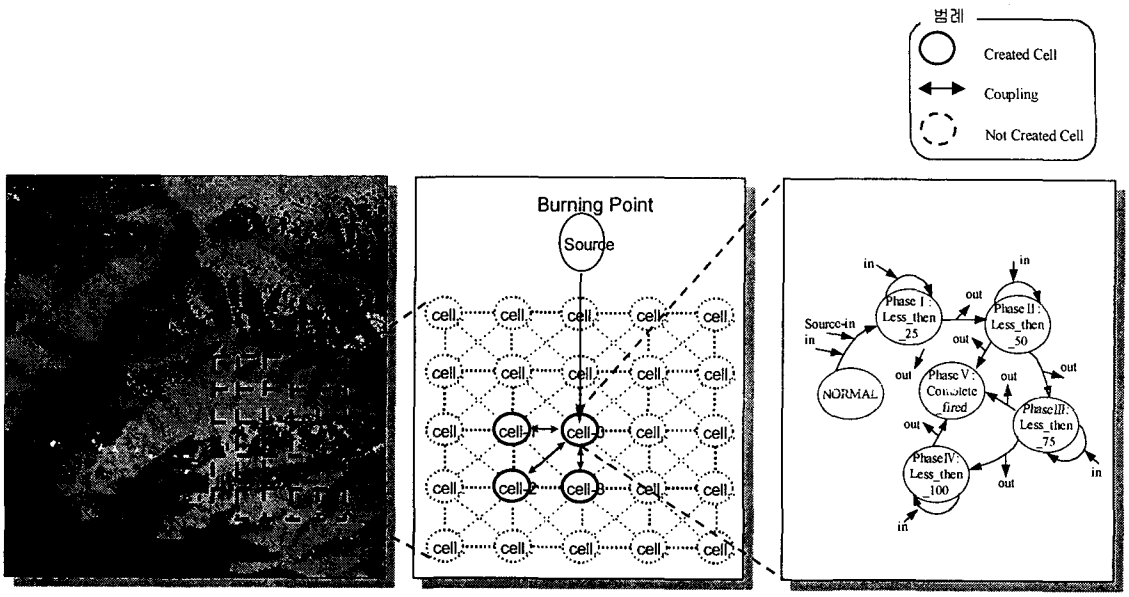
관들로부터 얻어지는 기상, 지형 및 산림정보 등의 데이터베이스를 이용하여 시뮬레이션을 통한 확산 예측 정보와 각종 모니터링 정보를 제공한다. 이를 위하여 그림8과 같이 가변구조형 셀기반의 모델링 접근방법을 제시하고 구현하였다. 먼저 대상 산림 지역을 셀별로 분할한 후, 첫째 각 셀별 지형, 기상, 산림 정보들의 데이터 모델링, 둘째 셀별 텍스쳐 매핑 및 3-D 그래픽 처리를 위한 공간 정보 모델링, 그리고 셋째 산불 확산 시뮬레이션을 위한 셀 단위의 동역학 모델링이 진행된다. 이러한 각 모델들은 산불 모니터링, 산림 정보, 기상 정보, 지형 정보, 가상 탐사, 그리고 산불 확산 예측 등 다양한 정보를 제공하는데 효과적으로 활용될 수 있다.

가변구조형 셀 모델링 접근 방법에 있어서, 셀 하나가 인접한 셀들에 미치는 영향력은 자신의 동역학적 행위의 결과에 의해 결정되어지며, 이러한 기능은 셀 모델 설계에서 더 많은 유연성과 효율성을 제공할 수 있다[8,9,10]. 가변구조형 산불모델링의 알고리즘은 다음과 같이 요약된다.

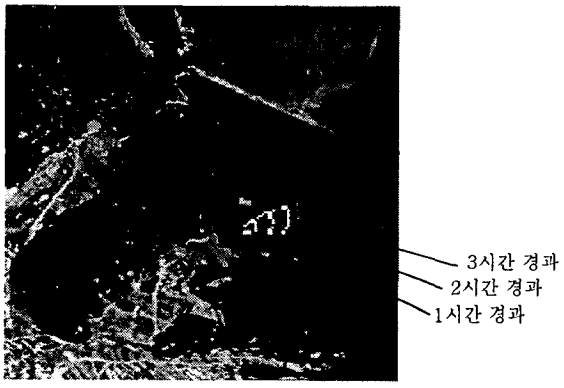
(1) 셀 모델링 및 모델베이스 구축.
(2) 외부로부터의 입력사건에 의해서 최초의 셀 모델이 생성 (산불 발생).
(3) 생성된 셀들의 동역학적 행위의 결과로 인접 셀 모델들이 생성.
(4) 셀 모델간의 결합과 상호작용 (산불이 번짐).
(5) 생성된 모든 셀들의 행위가 종료될 때까지 (3), (4)를 반복.

산불 확산 예측을 위한 시뮬레이션 모델은 산불 발생시 발화점이 되는 Source모델과 단위 지역별 동역학을 나타내는 셀 모델로 구성된다. 그림9은 가변구조 모델링을 사용한 셀기반 DEVS 모델링 방법을 나타낸다. 여기에서 Source모델에 의해 최초로 산불이 전달되어지는 Cell-0모델이 가변구조에 의해 먼저 생성되어진다. 이어서, Cell-0은 시간경과에 따른 자신의 상태 변환에 따라 다시 인접한 Cell-1, Cell-2, Cell-3에게 산불을 출력(인접 지역으로 산불이 번짐)할 수 있으며, 이때 비로써 Cell-1, Cell-2, Cell-3이 생성, Cell-0과 결합되어 새로운 시뮬레이션 구조로 변환된다. 여기서 셀 모델은 여섯 단계의 상태 값을 갖는 것으로 정하였고 각 상태는 자신이 갖게 되는 산불의 양과 자신의 상태에 따라 이웃 셀들과의 계속적인 상호관계 속에서 상태 변환을 일으키도록 모델링하였다. 즉, 각 셀은 인접 셀들로부터의 입력 사건과 자신의 상태에 따라서 지속시간인 연소속도와 인접 셀로의 불의 확산 양이 결정되고 그에 따라 인접 셀들로 출력을 내어 보내게 된다. 셀 모델의 상태는 초기상태인 Normal상태와 연소의 정도에 따라 Less_then_25, Less_then_50, Less_then_75, Less_then_100, Complete_fired 등이 있으며 각 상태에서 밀도로 나타내어지는 수목의 양은 상태가 변화될 때마다 20씩 감소되어 수목의 양이 0이 되면 Complete_fired 상태로 변환되고 불이 꺼지게 된다. 셀 모델의 상태 변화에 영향을 미치는 변수들에는 풍속, 경사도, 습도, 화력, 온도, 산림밀도, 수령, 수종 등이며, 각 변수는 그 값에 따라 각각 High, Middle-High, Middle, Middle-Low, 그리고 Low의 다섯 레벨로 나뉘어지고 각 변수들은 산불 확산에 미치는 영향력에 따라서 각각 고유의 가중치가 부여된다. 셀 모델의 연소속도와 인접 셀로 확산되는 산불의 양은(풍속의 Level × 가중치1)+(경사도의 Level × 가중치2)+.....+(수종의 Level × 가중치n)로 얻어진 값에 풍향과 경사방향과의 관계가 함께 고려되어서 결정된다[12]. 산불정보시스템에서 시간별로 산불이 진행되어 가는 과정은 <그림 10>과 같다.

<그림 11>은 각각의 시뮬레이션 범위에 대하여 실제로 시뮬레이션 하는데 걸리는 시간과 생성되는 셀의 개수를 통하여 가변구조와 고정구조의 성능을 비교한 것이다. 고정구조는 정해진 크기만큼의 셀을 시뮬레이션 하기 전에 모두 생성을 한 후에 시뮬레이션을 수행하기 때문에 처음에는 하나의 셀로 시뮬레이션을 수행하고 필요한 만큼만 셀을 생성하는 가변구조에 비해서 크기가 커지면 커질수록 실제로 시뮬레이션을 하는데 걸리는 시간이 훨씬 오래 걸리는 것을 볼 수 있다. 이러한 가



<그림 9> 가변구조를 이용한 산불 확산 모델링

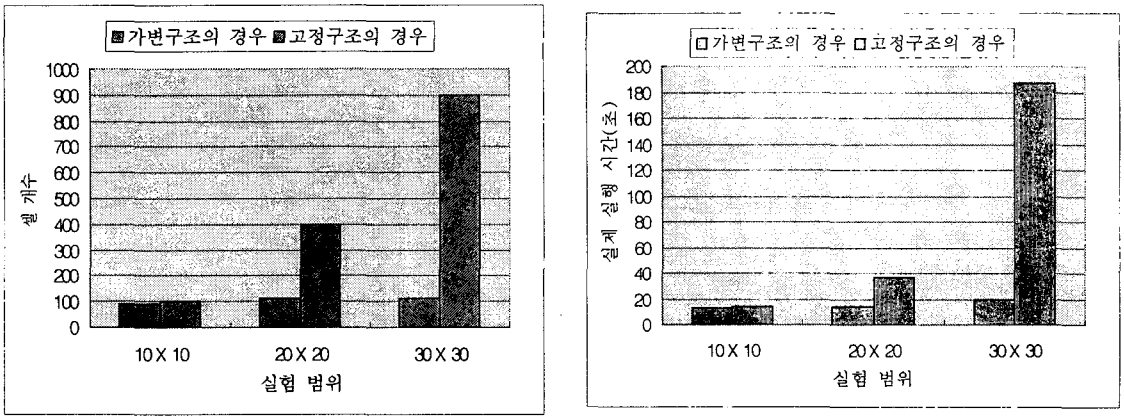


<그림 10> 시간별 산불 진행 상황

변구조 시스템에서는 구조가 변경될 때 구조 변경 메시지와 모델이 생성되고 소멸되는데 따른 overhead가 발생되지만 고정구조에서 모델이 다른 모델의 상태를 확인하는 메시지를 주고받는데 발생하는 overhead와 초기 모델들을 생성하는데 필요로 하는 overhead에 비하면 무시할만한 정도의 결과를 얻었다. 이와 같이 가변구조를 이용한 산불정보시스템은 산악지형을 셀단위의 공간으로 분할하고, 셀간의 결합관계는 가변구조를 적용함으로써 시스템 설계상의 효율성과 유연성을 제공한다.

5. 결론

본 논문은 DEVS 모델링 및 시뮬레이션 프레임워크가 갖는 고정된 구조로 인한 비효율성을 극복하기 위하여 4개의 가변구조 메세지를 도입함으로써 가변적인 환경에 적절하게 대처할 수 있는 가변구조 모델링 및 시뮬레이션 프레임워크의 설계와 구현을 주목적으로 하였다. 기존의 가변구조에 대한 연구가 다쪼각 개념과 다계급 개념 그리고 지능적 구조화 처리기 개념을 이용한 것과는 달리, 본 논문에서는 DEVS 형식론을 바탕으로 기존의 고정구조모델에 가변구조를 위한 새로운 클래스와 가변구조 연산자를 제안함으로써 가변구조형 모델링 및 시뮬레이션 S/W 환경인 D^2-SIM를 구현하였다. 또한, 다중프로세서 모델링의 사례연구를 통하여 적



(a) 생성된 셀 개수 비교 (b) 시뮬레이션 실행 시간 비교

<그림 11> 가변구조와 고정구조의 성능 비교

용 가능성을 검토하였고, 산불정보시스템을 위한 격자형 공간 모델링의 사례연구를 통하여는 가변구조에 대한 보다 실질적인 적용으로 고정구조 모델링과 가변구조 모델링의 실제 시뮬레이션 시간과 생성되는 모델 개수의 비교를 통해서 가변구조의 효과를 확인할 수 있었다. 제안된 가변구조 기법을 적용함으로써 시스템의 구조적 설계 상에 더욱 효율적이고 능동적으로 대처를 할 수가 있을 것으로 기대되며 향후 연구방향으로는 다양한 실험조건을 고려한 성능평가와 더불어 D^2-SIM 클래스 구조의 최적화에 대한 연구와 가변구조출력집합(Y_{vs})에 coupling 정보를 포함시킴으로써 보다 유연성 있는 구조변화에 대한 연구가 수행되어야 할 것이다.

참고 문헌

[1] B.P. Zeigler, "Object-Oriented Simulation with Hierarchical, Modular Models", *Academic Press*, SanDiego, CA, USA: 1990.

[2] B.P. Zeigler, "Toward a Simulation Methodology for Variable Structure Modelling", *In Modelling and Simulation Methodology in the Artificial Intelligence Era(eds: M.S. Elzas, T.I. Oren, B.P. Zeigler), North Holland Pub. Co, Amsterdam*, pp195-210, 1986.

[3] B.P. Zeigler and H. Praehofer, "Systems Theory challenges in the Simulation of Variable Structure and Intelligent Systems", *In: CAST-Computer-Aided Systems Theory, Lecture Notes, Springer-Verlag*, Berlin(in press), 1989.

[4] H. Praehofer, "An Environment for DEVS -based Multiformalism Modeling and Simulation in C++" *Published in: AIS 96, IN IEEE C/S*, San Diego, 1996

[5] F.J. Barros, "The Dynamic Structure Discrete Event System Specification Formalism", *Trans. of Computer Simulation International*, Vol 13, No.1 pp 35-46, Mar, 1996.

[6] F.J. Barros, "Dynamic Structure Discrete Event System Specification : Structure Inheritance in the DELTA Environment", *AIS'96 Sixth Annual Conference on AI, Simulation and Planning in High Autonomy Systems*, pp 141-147, Mar, 1996.

[7] B.P. Zeigler, "Concepts For Distributed Knowledge Maintenance In Variable Structure Models", *In MODELING AND SIMULATION METHODOLOGY, Knowledge Systems' Paradigms (eds: M.S. Elzas, T.I.Oren, B.P. Zeigler), Elsevier Science Publishers B.V (North-Holland)*, pp45-54, 1989

[8] B.P. Zeigler, "Multifaceted Modeling and Discrete Event Simulation", *Academic Press*, 1984

[9] B.P. Zeigler, "Theory of Modelling and Simulation", New York. *NY:Wiley*. 1976.

[10] Cho, T.H., Chi, S.D., "Name-Directed Coupling Applied to Cellular Model : River Pollution Example", *Int. Conf. On MODSIM 95*, Newcastle, Australia, Oct. 1995.[9]

[11] Y.K. Kim, "Model-based Design for Intelligent Systems : Intelligent Card Game Player", *Ma. Dissertation, Hankuk Aviation Univ*, pp 15-17, 1997.

[12] 지승도, 황수찬, 이종근, 이장세 외, "FOFIS : 산불 정보 시스템", *한국시뮬레이션 학회 논문지* 제8권 2호, 1999.

[13] 이장세, 이종근, 황수찬, 지승도, '산불정보 시스템을 위한 격자형 공간 모델링 및 시뮬레이션", *정보과학회 춘계학술발표논문집*, 4. 1998

부록. DELTA 시스템과의 비교

본 부록에서는 가변구조에 관한 기존 연구 중의 하나로서 DSDEVS(Dynamic Strcuture Discrete Event System Specification)형식론[5,6]을 통하여 구현된 DELTA 시스템과 본 연구에서 개발한 D^2-SIM과의 특성을 비교분석한다.

1) DELTA 시스템(DSDEVS) 개요

DSDEVS 형식론은 시스템의 동적인 구조를 표현하기 위해 모델의 표현자체를 네크워크 구조로 구성하였다는 특징이 있다. 이에 따른 DSDEVS 형식론은 기본 모델과 동적 구조를 갖는 네트워크 모델로 구성된다.

기본 모델

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

네트워크 모델

$$DSDEVS_{\wedge} = <X, M_X>$$

$\triangle$ = DSDEVS 네트워크 이름.
X = DSDEVS 네트워크 실행 이름.
M_X = X의 모델.

또한, 네트워크 모델은 네트워크 실행 X라는 특별한 요소이자 실행 모델인 M_X 의해서 정의될 수 있다.

$$M_X = < X_X, S_X, Y_X, \delta_{intX}, \delta_{extX}, \lambda_X, ta_X>$$

특히 상태집합 $s_X \in S_X$ 은 DSDEVS 네트워크 구조에 대한 정보를 가지며 다음과 같은 8개의 구성요소로 이루어진다.

$$s_X = < X^{X}_{\triangle}, Y^{X}_{\triangle}, D^{X}_{\wedge}, \{M^{X}_{i}\}, \{I^{X}_{i}\}, \{Z^{X}_{i,j}\}, \Xi^{X}, \theta^{X}>$$

$X^{X}_{\triangle}$ = DSDEVS 네트워크의 입력 집합.
$Y^{X}_{\triangle}$ = DSDEVS 네트워크의 출력 집합.
D^{X} = 요소들의 집합.
M^{X}_{i} = i 요소의 모델, 모든 $i \in D^{X}$.
I^{X}_{i} = i의 영향력, 모든 $i \in D^{X} \cup \{X,\triangle\}$.
$Z^{X}_{i,j}$ = i-to-j의 변역 함수, 모든 $j \in I^{X}_{i}$
Ξ^{X} = 선택 함수.
θ^{X} = 과거 정의되지 않은 다른 상태의 변수.

이와 같이 네트워크 모델의 실행모델이 모든 조합과 연결에 관한 정보를 가짐으로써 동적인 구조를 표현할 수 있는 방법을 제공하게 된다.

<표 1> D^2-SIM과 DELTA의 형식론 비교

구분	D^2-SIM	DELTA
기본 모델	$M_{var} = <X,S,Y,\delta_{int},\delta_{ext},\lambda,Y_{vs},\lambda_{vs},ta>$	$M = <X,S,Y,\delta_{int},\delta_{ext},\lambda,ta>$
결합 모델 (네트워크 모델)	$DN = <D,\{M\},\{I\},\{Z\},select>$	$DSDEVS_{\wedge} = <X, M_X>$ $M_X = <X_X, S_X, Y_X, \delta_{intX}, \delta_{extX}, \lambda_X, ta_X>$ $s_X = <X^X_{\wedge},Y^X_{\wedge},D^X_{\wedge},\{M^X_i\},\{I^X_i\},\{Z^X_{ij}\},D^X,D^X>$

2) D^2-SIM과 DELTA 비교

D^2-SIM과 DELTA의 차이는 다음과 같다. 기본 모델에 있어서 D^2-SIM은 DEVS 형식론의 기본 모델에 가변구조함수와 가변구조 집합을 추가하여 가변구조를 구현한데 반해 DELTA는 DEVS 형식론의 기본 모델에서는 동일한 형태를 가진다. 또한, 결합 모델에 있어서도 D^2-SIM은 DEVS 형식론의 결합 모델과 동일한 형태를 가지는 반면 DELTA는 DEVS 형식론의 결합 모델과 달리 동적 구조 정보를 가지는 네트워크 모델이 존재한다. D^2-SIM은 DEVS 형식론에서 가변구조를 구현할 수 있는 집합과 함수만 포함시킴으로써 기본적으로 DEVS 형식론이 나타낼 수 있는 모든 구조를 제한 없이 표현할 수 있을 뿐 아니라 가변구조도 표현할 수 있지만 DELTA모델은 네트워크 모델이라는 변형된 형식론을 적용함으로써 일반적이 아닌 특별한 형태로 제한을 하였기 때문에 표현의 한계를 가지고 있다.(<표 1>)

다음 <표 2>는 D^2-SIM과 DELTA 시스템의 주요 특징을 비교한 것으로써 첫째, D^2-SIM은 원자 모델에서 직접 구조변화에 대한 요구의 감지를 하는 반면에 DELTA 시스템은 다른 모델들을 항상 모니터링하는 제어모델을 필요로 한다. 둘째, D^2-SIM은 결합정보를 DEVS 형식론과 동일한 결합모델이 가지고 있는 반면에 DELTA 시스템은 실행모델이 결합정보를 가지고 있으면서 가변구조에 대한 모든 제어를 하게 된다. 셋째, 모델링의 편의성에 대해서는 D^2-SIM은 DEVS 모델링의 형식론에 가변구조에 대한 함수와 집합을 추가함으로써 가변구조를 위한 추가적인 모델을 생성할 필요가 없으므로 모델링이 용이한 반면 DELTA 시스템은 가변구조를 위해서 추가적인 모델(실행모델, 제어모델)을 항상 생성함으로 시스템이 복잡해진다. 마지막으로 D^2-SIM은 초기구조만 가지만 어떠한 형태로든지 그 구조를 변화시킬 수 있지만 DELTA시스템은 실행 모델이 가지고 있는 구조 변화에 대한 방법 내에서 그 구조를 변화시킬 수 있기 때문에 구조 변화에 대한 한계를 가지게 된다.

<표 2> D^2-SIM과 DELTA의 특징 비교

구분	D^2-SIM	DELTA
구조변화 요청	원자 모델에서 구조변화를 직접 요청.	제어 모델이 다른 모델의 상황을 항상 모니터링 한 후에 결정.
결합 정보	결합 모델이 가짐.	실행 모델이 가짐.
추가적 모델링 필요성	가변구조를 위해 추가적인 모델이 생성 불필요.	가변구조를 위해 추가적인 모델(실행 모델)이 필요.
구조변화의 한계성	초기 모델링 조건에 따라 계속적으로 구조를 변화시킬 수 있음.	구조 변화 방법이 사전에 정해져 있으므로 구조 변화의 한계성을 갖음.

저자소개

지승도

1982년 연세대학교 전기공학과 졸업(공학사)
1984년 연세대학교 대학원 전기공학과 졸업(공학석사)
1985년~1986년 두산 컴퓨터(현 한국 디지털)근무
1991년 미국 아리조나대학교 전기전산공학과 졸업(공학박사)
1991년~1992년 미국 SIMEX Systems and S/W 회사 S/W담당자로 근무
1992년~현재 한국항공대학교 컴퓨터공학과 부교수
관심분야 : 지능시스템 디자인 방법론, 교통모델링, 이산사건 시스템 모델링 및 시뮬레이션, 시뮬레이션 기반 인공생명, 컴퓨터 보안 등임

이종근

1996년 한국항공대학교 전자계산학과 졸업(이학사)
1998년 한국항공대학교 대학원 컴퓨터공학과 졸업(공학석사)
1998년~현재 한국항공대학교 대학원 컴퓨터공학과 박사과정
관심분야 : 지능시스템 디자인 방법론, 교통모델링, 이산사건 시스템 모델링 및 시뮬레이 션, 자치적 능동 방어시스템, 컴퓨터 보안 등임

이장세

1997년 한국항공대학교 전자계산학과 졸업(이학사)
1999년 한국항공대학교 대학원 컴퓨터공학과 졸업(공학석사)
1999년~현재 한국항공대학교 대학원 컴퓨터공학과 박사과정
관심분야 : 지능시스템 디자인 방법론, 교통모델링, 이산사건 시스템 모델링 및 시뮬레이 션, 시뮬레이션 기반 인공생명, 컴퓨터 보안 등임

정기찬

1999년 한국항공대학교 컴퓨터공학과 졸업(공학사)
1999년~현재 한국항공대학교 대학원 컴퓨터공학과 석사과정
관심분야 : 지능시스템 디자인 방법론, 교통모델링, 이산사건 시스템 모델링 및 시뮬레이 션, 가변구조 모델링, 지능제어 등임.