DEVS 모델과 사용자 요구사항의 일관성 검증 방법론 및 환경 구현

김도형*, 김탁곤**

Method and Implementation for Consistency Verification of DEVS Model against User Requirement

Do-Hyung Kim, Tag-Gon Kim

Abstract

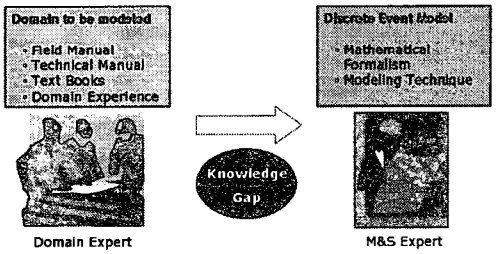
Development of complex discrete event simulators requires cooperation between domain experts and modeling experts who involve the development. With the cooperation the domain experts derive user requirement and modeling experts transform the requirement to a simulation model. This paper proposes a method for consistency verification of simulation model in DEVS formalism against the user requirement in UML diagrams. It also presents an automated tool, called VeriDEVS, which implements the proposed method. Inputs of VeriDEVS are three UML diagrams, namely use case, class and sequence diagrams, and DEVS Graph, all in Visio; outputs of a verification result is represented in PowerPoint files.

Key Words: Consistency Verification, DEVS model, User Requirement, UML diagram, Automated tool, VeriDEVS

* 국방과학연구소 4본부 연구원
** 한국과학기술원 전자전산학과 교수

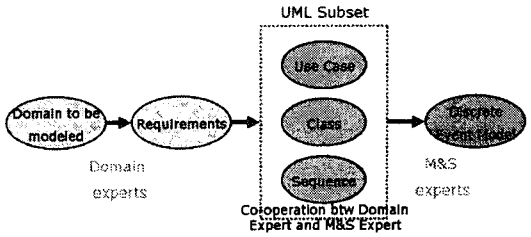
1. 서론

이산사건 시뮬레이터는 대상 시스템을 잘 관찰하여 객체 지향적인 방법을 통해 모델을 만든 뒤, 모델을 시뮬레이션 할 수 있게 하는 도구이다. 따라서 실제 시스템의 동작과 일치하는 정확한 시뮬레이터를 개발하기 위해서는 대상 시스템에 대한 정확한 이해를 바탕으로 이산사건 모델을 개발해야 한다. 그러나 모델링 도메인의 영역이 세분화되고 복잡해짐에 따라 도메인에 대한 전문적인 지식을 가진 도메인 전문가와 모델링을 수행하는 모델링 전문가 사이의 의사소통은 점점 더 어려워지고 있다. <그림 1>에서 나타난 것처럼 군사용 워게임 모델 개발과정에서 도메인 전문가는 군사용 교범에 익숙하고 실전 경험이 풍부한 군인이 되고, 모델링 전문가는 수학적인 형식론에 익숙하고 풍부한 모델링 경험을 가진 M&S(modeling and simulation) 기술자가 될 것이다. 이산사건 모델 개발을 위해서 군인과 M&S 기술자가 반복적인 의사소통을 해야 하지만, 두 전문가들은 상호 영역에 대한 지식의 부족으로 인해 원활한 협력 작업이 어렵다.



<그림 1> 이산사건 모델 개발 과정

이러한 문제를 해결하기 위해 도메인 전문가와 모델링 전문가 모두 쉽게 이해할 수 있는 의사소통 도구를 제시하여 효율적인 이산사건 모델 개발을 할 수 있도록 하는 방법이 제시되었다.[1] <그림 2>는 새롭게 제시된 이산사건 모델 개발 과정을 자세히 나타내고 있다. 이 방법에서는 시스템의 기능적 요구사항들을 정리한 사용자 요구명세서로부터 DEVS 모델을 개발하는 중간 단계에 UML 다이어그램을 사용하여 정확하고 간단한 이산사건 모델 개발 과정을 제시하고 있다.



<그림 2> UML을 이용한 이산사건 모델 개발 과정[1]

1.1 연구 동기

M&S 전문가에 의해 개발된 이산사건 모델은 도메인 전문가가 작성한 사용자 요구사항의 내용과 일치해야 한다. 만일 사용자 요구사항과 일치하지 않은 모델을 설계하였을 경우, 그 모델에 대한 시뮬레이션의 결과는 사용자에게 아무런 의미를 가질 수 없다. 따라서 최종 개발된 DEVS 모델에 대해서 요구명세서의 내용과 일치하게 개발되었음을 증명할 필요가 있다. 특히 대상 시스템이 크고 복잡할수록 모델의 유효성 검증이 어렵기 때문에 개발 과정에서 점진적으로 수행할 수 있는 체계적인 검증 방법이 필요하다.

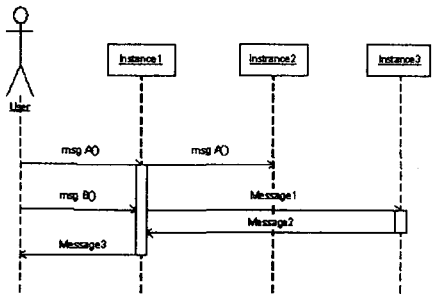
따라서 본 논문에서는 앞서 소개한 UML을 중간단계로 이용한 DEVS모델 개발 방법에 따라 설계된 이산사건 모델을 체계적이고 자동으로 검증하는 방법을 제안한다. 전체 검증과정은 개발과정에서 산출된 UML 다이어그램과 사용자 요구사항간의 일관성 검증과 UML 시퀀스 다이어그램[3]과 DEVS 모델 사이의 이벤트 시퀀스 검증으로 이루어진다. 그리고 제안한 검증 방법을 자동적으로 수행할 수 있는 소프트웨어 도구(VERIDEVS)를 제공한다. VERIDEVS는 Visio를 이용해 도식된 UML 다이어그램과 DEVS 그래프를[5] 입력으로 하여 제안한 검증 과정을 자동 수행한 후 그 결과를 본 논문에서 제안하는 검증테이블 형태로 출력한다.

2. 2장 UML과 DEVS 형식론

2.1 UML (Unified Modeling Language) [2]

UML은 오늘날의 객체 지향 시스템 개발 분야에서 가장 주목 받는 도구 중 하나이다. UML은 시스템 개발자가 자신의 시야를 구축하고 반영하는데 있어서 표준적이고 이해하기

쉬운 방법으로 할 수 있도록 도와주며 자신의 설계 결과물을 다른 사람과 효과적으로 주고받으며 공유할 수 있는 메커니즘을 제공하기 때문이다. 앞서 말한 이산사건 모델 개발 과정에서는 UML의 여러 다이어그램 중 사용자의 입장에서 바라본 시스템의 특성을 설명한 유스케이스 다이어그램, 사물이 가진 속성과 일정한 행동 수단을 기술하는 클래스 다이어그램, 그리고 시스템 내의 각 객체들이 시간의 흐름에 따른 교류 상황을 나타내는 시퀀스다이어그램의 세 가지를 이용하여 사용자 요구사항을 UML 다이어그램으로 나타낸다.



<그림 3> 시퀀스 다이어그램

<그림 3>은 시퀀스 다이어그램의 예를 나타내는 것으로, 시퀀스 다이어그램은 시스템을 구성하는 객체를 나열하고 아래 방향으로 시간을 진행하며 객체들 사이에 이루어지는 교류를 나타낸다.

2.2 DEVS 형식론 [4]

이산 집합론에 근거한 DEVS 형식론은 이산사건 시스템을 모듈 별로 나누고 이를 계층적인 연결로 모델링 할 수 있는 수학적인 기반을 제공하는 모델링 방법론이다. DEVS에서는 원자 모델(Atomic Model)과 결합 모델(Coupled Model)을 정의한다. 원자 모델은 DEVS 형식론을 구성하는 가장 기본적인 모듈로서 시스템의 행동을 기술하는 모델이다. 원자 모델은 DEVS 형식론에 기반을 두어 다음과 같이 세 가지의 집합과 네 개의 함수로 수학적 표현이 된다.

$AM = \langle X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta \rangle$

X: 이산사건 입력 집합

Y: 이산사건 출력 집합

S: 일련의 상태의 집합

δ_{int}: $Q \rightarrow Q$: 내부 상태 천이 함수

$Q = \{(s, e) | s \in S, 0 \le e \le ta(s)\}$: total state of M

δ_{ext}: $Q * X \rightarrow S$: 외부 상태 천이 함수

λ: $Q \rightarrow Y$: 이산사건 출력 함수

ta: $S \rightarrow R_{0,\infty}^{+}$: 시간 진행 함수

결합 모델은 시스템 구성 요소간의 상호 작용을 표현하기 위한 모델로서 시스템 내부의 컴포넌트와 외부 입력, 외부 출력 사이의 연결 정보를 표현한다. UML 다이어그램을 이용한 이산사건 모델 개발 방법에 따라 DEVS 모델을 개발할 때, 최종 DEVS 모델은 시퀀스 다이어그램의 입력 이벤트, 출력 이벤트, 조건문, 그리고 클래스에서 수행되는 액티비티(Activity) 정보를 기초로 개발된다. 그러나 앞서 설명한 기본 원자 모델에서는 각 상태에서 수행되는 액티비티에 대한 정보를 나타낼 방법이 없다. 따라서 각 상태에서 수행되는 액티비티를 정의하기 위해 Real-Time DEVS(RT-DEVS)를[6] 참고하여 액티비티 집합과 액티비티 매핑(mapping)함수를 정의하였다.

$eAM = \langle X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta, A, \Psi \rangle$

X, Y, S, δ_{int}, δ_{ext}, λ, ta: same as classic DEVS

A: 액티비티 집합

Ψ: $S \rightarrow 2^{A}$: 액티비티 매핑 함수

3. DEVS 모델 검증 방법론

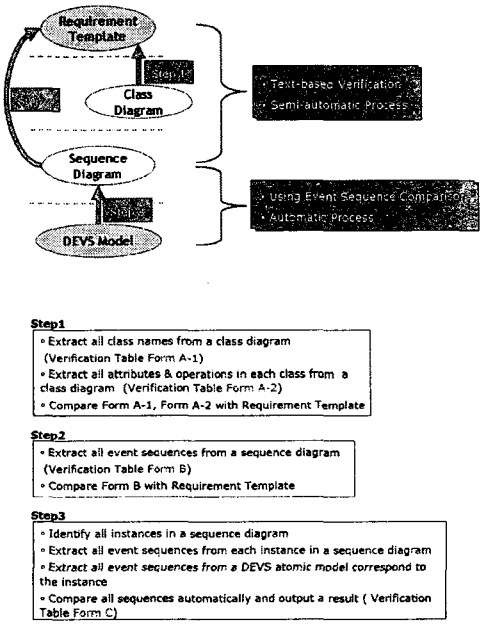
3.1 검증 방법론 개요

제안된 검증 방법은 개발 방법을 비추어 볼 때, 세 단계의 점진적인 검증으로 나눌 수 있다. 즉, 개발 과정에서 사용자 요구사항으로부터 DEVS 모델 개발 과정에서 만들어지는 UML 다이어그램을 사용하여 부분 검증을 한다. <그림 4>는 제안하는 검증 방법의 전체 과정을 간단히 나타낸 것으로, 이산사건 모델의 검증은 사용자 요구사항과 UML 클래스 다이어그램, 사용자 요구사항과 UML의 시퀀스 다이어그램, 그리고 시퀀스 다이어그램과 DEVS 모델 사이의 검증으로 이루어진다.

3.2 Step 1: 클래스 다이어그램과 사용자 요구명세 검증

UML의 클래스 다이어그램은 시스템에 참여하고 있거나 시스템에 의해 사용되는 객체 타입, 즉 클래스들을 정의하고 클래스들 간에 존재하는 정적인 관계를 다양한 방식으로 표현한 다이어그램이다. 클래스 다이어그램에서 사용자 요구 명세서와 검증해야 할 부분은 다음과 같다.

1. 시스템의 정적 구조를 나타내는 기본 단위인 클래스 종류와 클래스 식별에 기준이 된 근거
2. 각 클래스가 지닌 속성
3. 각 클래스가 가지는 오퍼레이션



<그림 4> DEVS 모델 검증 방법 구성

본 논문에서는 클래스 다이어그램에 나타나 있는 모든 클래스 정보를 나열할 수 있는 검증 테이블을 제안한다. 제안한 클래스 검증 테이블은 검증해야 할 대상을 체크 리스트 형태로 명시적으로 표현하기 때문에, 검증 대상을 명확히 하고 클래스 정보의 손실 여부를 쉽게 판별할 수 있게 한다. 클래스 다이어그램 검증 테이블은 클래스 다이어그램에서 검증되어야 할 내용에 맞추어, 두 가지 형태의 테이블(Form A-1: Class Identification Table, Form A-2: Class Table)로 구성되어 있다.

<표 1> Class Identification Table (Form A-1)

Part	①,	goal	②,
Basis,	Class name,	User Requirements,	
③,	④,	⑤,	
…,	…,	…,	
…,	…,	…,	
…,	…,	…,	

<표 2> Class Table (Form A-2)

Part	⑥,	goal	⑦,
Class Diagram,		User Requirements,	
Name,		⑧,	
⑨,		…,	
Attributes,		…,	
⑩,		…,	
Operations,		…,	
⑪,		…,	

3.3 Step 2: 시퀀스 다이어그램과 사용자 요구명세 검증

UML의 시퀀스 다이어그램은 클래스 사이의 상호 작용을 시간에 따라 나타낸 다이어그램이다. 각 시퀀스 다이어그램은 시스템의 여러 유스 케이스 중 하나의 흐름을 나타낸다. 따라서 시퀀스 다이어그램과 요구명세서의 검증에서는 하나의 유스 케이스에 대한 이벤트 시퀀스를 사용자 요구 명세의 내용과 일치를 확인한다. 시퀀스 다이어그램의 검증을 위해서는 아래와 같은 내용이 요구명세서의 내용과 일치하는가를 맞추어 본다.

1. 유스 케이스에 참여하고 있는 객체의 종류
2. 객체로부터 생성되는 메시지의 종류와 메시지를 수신하는 객체
3. 시간 순서에 따른 객체들 사이의 메시지 상호 작용과 각 객체에서의 액션 정보

시퀀스 다이어그램의 검증 역시 요구명세서의 이벤트 시퀀스 정보와 일치하는 부분을 검색하여 비교하는 과정으로 이루어진다. 클래스 다이어그램과 마찬가지로 시퀀스 다이어그램 검증을 위한 체크리스트 형태의 검증 테이블 Form B를 제공하여 정확한 비교를 할 수 있다.

<표 3> Sub-Sequence Table (Form B)

Event	①	Event	②
Sequence Diagram		User Requirements	
If	Then		
⑦	④	⑤	
...	...	...	
...	...	...	
...	...	...	

3.4 Step 3: DEVS 모델과 시퀀스 다이어그램 검증

본 단계에서는 시퀀스 다이어그램의 이벤트 시퀀스와 DEVS 모델의 이벤트 시퀀스를 검증한다. 우선 양쪽 설계 명세로부터 비교되어야 할 모든 이벤트 시퀀스를 추출하여 상호 대응되는 이벤트를 자동 비교하고 결과를 검증테이블 Form C로 출력함으로써 검증이 완료된다.

```
Algorithm EventSequenceTreeConstruction
    Input : G=(V, E)
    Output: ST = (V_ST, E_ST)
    begin
    DFST = (V_DFST, E_DFST) = dfst(G)
    V_ST += V_DFST
    E_ST += E_DFST
    BWE = bwe(G)
    for each edge=(V1, V2) in BWE do
        V_ST += dup(V2)
            E_ST += (V1, dup(V2))
    end
end
```

```
dfst(G) : return a depth first spanning tree of an input graph G
bwe(G) : return all backward edges of an input graph G
dup(V) : return a duplicated node of an input node V
```

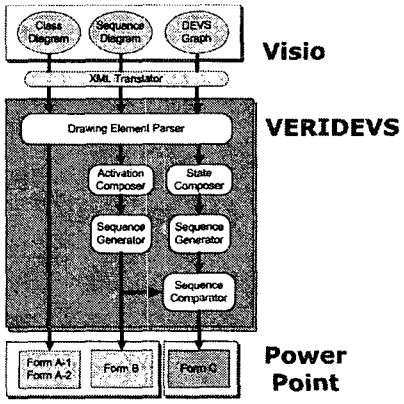
<그림 5> 이벤트 시퀀스 트리 구성 알고리즘

<표 4> 이벤트 시퀀스 테이블 (Form C)

Event	①	Event	②
N	Sequences		V_n
⑥	④		⑤
...	...		...
...	...		...
...	...		...

3. VERIDEVS: DEVS 모델 자동 검증 환경

VERIDEVS는 제안한 검증 방법을 자동적으로 수행할 수 있는 환경이다. VERIDEVS가 DEVS 모델 검증을 자동적으로 수행하는 기본 흐름은 <그림 6>과 같다.



<그림 6> VERIDEVS 구조

우선 검증해야 될 대상이 되는 UML 다이어그램과 DEVS 모델은 Visio를 이용하여 Visio 드로잉으로 도식된다. 도식된 Visio 드로잉은 Visio에서 제공하는 XML(Extensible Markup Language)변환기를 통해 XML 언어로 변환 후 VERIDEVS의 입력으로 들어간다. 이러한 입력으로부터 VERIDEVS는 클래스 인식, 시퀀스 다이어그램과 DEVS 그래프로부터 시퀀스를 추출하고 양쪽의 이벤트 시퀀스를 서로 비교한다. 마지막으로 VERIDEVS에 의해 분석된 모든 정보는 앞서 제안한 검증 테이블을 MS PowerPoint의 형태로 사용자에게 출력된다. VERIDEVS의 주요 컴포넌트를 살펴보면, 입력으로 들어오는 XML 언어로부터 드로잉의 기본 Element를 분석하는 Drawing Element Parser가 있다. 기본 Element를 이용해 Activation Composer는 시퀀스 다이어그램의 정보를 추출하고 State Composer는 DEVS 그래프의 정보를 추출하고, Sequence Generator를 통해 비교할 수 있는 모든 이벤트 시퀀스가 추출된다. 그리고 Sequence Comparator를 통해 양쪽 이벤트 시퀀스가 자동 비교되어 그 결과를 검증테이블로 출력함으로써 모든 검증이 종료된다.

4. 결론

본 논문에서는 UML 다이어그램을 중간 단계로 이용한 이산사건 시뮬레이터 개발 방법론을 통해 개발된 DEVS 모델에 대해, 사용자 요구명세서에 나타난 기능적 요구사항을 잘 반영하고 있는가에 대한 검증 방법을 제시하고 이를 자동으로 수행할 수 있는 VERIDEVS를 구현하였다. 본 논문에서 제안한 검증 방법은 DEVS 그래프로 표현된 DEVS 모델을 대상으로 하기 때문에, 그래프로 표현하기 어려운 모델에는 적용하기 힘들다. 따라서 DEVS 모델을 기술하는 텍스트 기반의 형태에도 본 검증 방법을 자동화 하는 것이 추후 과제로 남아있다.

참고문헌

[1] Su Youn Hong and Tag Gon Kim, "Embedding UML Subset into Object-oriented DEVS Modeling Process", *in Proc. SCSC*, San Jose', California, USA, pp. 161-166., July 2004,

[2] Mark Priestley, *Practical Object-Oriented Design with UML*, McGraw-Hill, 1996.

[3] Joseph Schmuller, *Teach Yourself UML in 24 Hours*, 2/E, SAMS, 2002.

[4] Bernard P.Zeigler, Herbert Praehofer, and Tag Gon Kim. *Theory of Modeling and Simulation*, ACADEMIC PRESS, 2000.

[5] Hong Sung Kim, "A Visual Modeling Environment Based on DEVS formalism", *M.S. Thesis*, KAIST, Daejeon, Korea, 1995.

[6] Hong, J. S., H. S. Song, "A Formal Method for Object-Oriented Real-time Software Development," *Ph.D. Thesis*, EE Dept., KAIST, Dec.1997