한국시뮬레이션학회 논문지
Vol. 21, No. 2, pp. 31-39 (2012. 6)

무기체계 교전 시뮬레이션을 위한 매트랩 기반 이산사건시뮬레이션 프레임워크의 개발

황근철[1] · 이민규[1†] · 김정훈[1†]

The Development of a MATLAB-based Discrete Event Simulation Framework for the Engagement Simulations of the Weapon Systems

Kun-Chul Hwang · Min-Gyu Lee · Jung-Hoon Kim

ABSTRACT

Simulation Framework is a basic software tool used to develop simulation applications. This paper describes the development of a discrete event simulation framework based on DEVS(Discrete EVent System Specification) formalism, using MATLAB language which is widely used in technical computing and engineering disciplines. The newly developed framework utilizing MATLAB object oriented programming combines the convenience of MATLAB language and the sophisticated architecture of the DEVS formalism. Hence, it supports the productivity, flexibility, extensibility that are required for the simulation application software development of the weapon systems engagement. Moreover, it promises a simulation application the increased the computation speed proportional to the number of CPU of a multi-core processor, providing the batch simulation functionality based on MATLAB parallel computing technology

Key words : Simulation Framework, DEVS(Discrete EVent System Specification) Formalism, MATLAB/SIMULINK, Weapon Systems, Parallel Comupting, Multicore Processor

요 약

시뮬레이션 프레임워크는 시뮬레이션 응용 프로그램의 개발을 지원하는 기반 소프트웨어이다. 본 논문은 공학용 프로그래밍 언어로 광범위하게 사용되는 매트랩을 이용하여 개발된 이산사건시뮬레이션 프레임워크의 개발 과정을 기술하고 있다. 매트랩 객체지향프로그래밍을 토대로 새롭게 개발된 프레임워크는 매트랩 언어의 편리성과 이산사건시뮬레이션 형식론(DEVS: Discrete EVent System Specification Formalism)이 가지는 뛰어난 개발 방법론을 결합시킴으로써 무기체계 교전 시뮬레이션 프로그램 개발에서 요구되는 생산성, 유연성, 확장성을 제공한다. 더불어 매트랩의 병렬컴퓨팅 기술을 적용한 배치(Batch) 시뮬레이션 기능을 제공함으로써 몬테카를로 시뮬레이션 수행시 컴퓨터 환경에서 지원되는 CPU 코어의 수에 비례하여 응용 프로그램의 연산성능을 향상시킨다.

주요어 : 시뮬레이션 프레임워크, 이산사건시뮬레이션 형식론(DEVS: Discrete EVent System Specification Formalism), 매트랩/시뮬링크, 무기체계 시뮬레이션, 병렬컴퓨팅, 멀티코어 프로세서

1. 서 론

시뮬레이션 프레임워크는 시뮬레이션 응용 프로그램을 개발하기 위해 제공되는 기반 소프트웨어로서, 재사용가능한 각종 라이브러리와 모델개발을 위한 클래스 API (Application Programming Interface), 그리고 통계분석 도구 및 가시화도구와 같이 응용 프로그램 개발을 지원/

접수일(2012년 5월 22일), 심사일(1차 : 2012년 5월 30일), 게재 확정일(2012년 5월 30일)
[1] 국방과학연구소 제6기술연구본부 1부
주 저 자 : 황근철
교신저자 : 이민규, 김정훈
E-mail; messin@add.re.kr, kimjh@add.re.kr

분석하기 위한 각종 툴박스로 구성된다[1]. 시뮬레이션 프레임워크는 소프트웨어 개발을 지원하는 단순 프로그램 라이브러리와 달리 시뮬레이션 응용 프로그램의 시뮬레이션 실행 원리(시간관리기능)와 모델 구조(데이터관리기능)를 결정하기 때문에, 시뮬레이션의 핵심 기능을 관장하는 의미에서 시뮬레이션 엔진 또는 시뮬레이션 개발 환경으로 불리 운다[2]. 이러한 시뮬레이션 프레임워크는 대부분 상용/비상용의 패키지 소프트웨어의 형태로 개발자에게 제공되는데, 그 수준에 따라 라이브러리 형태 또는 하나의 완성된 사용자 인터페이스 화면에서 모델의 작성/실행/분석을 모두 수행할 수 있는 통합개발환경(Integrated Development Environment) 형태로 구분된다. 대표적인 예로는 라이브러리 형태의 C-SIM, DEVSim++와 통합개발환경형태의 모델리카(Modelica), 매트랩/시뮬링크(Matlab/Simulink), 아레나(Arena), 애니로직(AnyLogic) 등이 있다[3].

언급된 시뮬레이션 프레임워크 중에서 매트랩/시뮬링크는 스크립트 방식의 손쉬운 프로그래밍 환경 및 블록다이어그램 기반의 편리한 그래픽 모델링 기법을 지원하는 모델기반 시뮬레이션 및 시스템 설계 도구로서, 공학 프로그래밍(Technical Computing) 및 동역학 해석/ 신호처리의 분야의 시뮬레이션 시장에서 2008년 기준 55%가 넘는 시장 점유율을 기록하고 있으며 항공/국방 체계 설계와 같은 특화된 응용분야에서도 폭넓은 사용자층을 확보하고 있다[4-6]. 이러한 매트랩/시뮬링크는 미분방정식과 차분방정식 같은 수식으로 표현되는 부체계 및 컴포넌트 단위의 모델링에서는 신속하고 편리한 시뮬레이션 환경을 제공하지만, 객체지향에 기반한 시뮬레이션 모델의 조립 및 동작정의에 대한 프로그래밍 기법과 도구를 제공하지 않았기 때문에 다수의 무기체계 객체(인스턴스)가 참여하는 상위수준의 체계 시뮬레이션을 개발하기에는 많은 제약 사항이 있었다[7].

본 논문은 무기체계 교전 시뮬레이션에서 요구되는 특성들 즉, 기동부/탐지부/제어부와 같은 여러 부체계로 구성되는 무기체계의 조립성과 무기체계의 운용과정에서 발생하는 다양한 교전 이벤트의 처리기능을 매트랩 프로그래밍 환경에 효과적으로 반영하기 위해, 새롭게 개발된 매트랩 이산사건 시뮬레이션 프레임워크에 대해 기술한다.

2. 이산사건 시뮬레이션 형식론

이산사건시뮬레이션(DEVS: Discrete Event Simulation) 형식론은 집합이론과 시스템 이론을 바탕으로 1976년 B. P. Zeigler에 의해 제안된 모델링 이론으로써 시뮬레이션 프로그램 개발에 있어 UML(Unified Modeling Language)과 디자인패턴의 역할을 수행하는 일종의 표준화 도구이다[8]. DEVS 다이어그램으로 표현되는 DEVS 모델의 명세는 UML과 매우 흡사한데, 이는 DEVS와 UML 모두 객체 지향적 사고에 근간을 두고 있기 때문이다. 그러나 DEVS는 UML이 표현하지 못하는 시뮬레이션 시관관리 기능을 모델 명세에 반영하고 있으며, 시뮬레이션에 특화된 디자인 패턴으로 계층적인 모델 구조와 독립된 시뮬레이션 통제 알고리즘을 제공하고 있다. DEVS 프레임워크는 이러한 DEVS 형식론이 실제 시뮬레이션 프로그램 개발에 적용될 수 있도록 지원하는 기반 소프트웨어로서 DEVS 형식론에 따른 모델 구성과 시뮬레이션을 실행 및 분석 할 수 있는 다양한 API(Application Programming Interface)를 제공하며, 대표적으로는 C++로 개발된 DEVSim++와 JAVA로 개발된 DEVSJAVA가 있다[9,10].

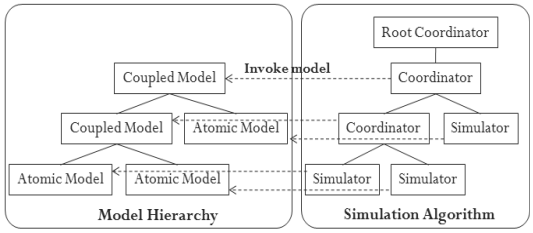


그림 1. DEVS 형식론의 모델 및 시뮬레이션 구조

DEVS 형식론은 집합이론과 시스템 이론을 바탕으로 한 수학적 명세로 시뮬레이션 모델을 정의한다. DEVS 형식론에서 핵심적으로 제공하는 계층적인 시뮬레이션 모델 구조와 모델 실행 순서를 스케쥴링 하는 시뮬레이션 알고리즘은 다음과 같다.

DEVS의 모델 구조는 그림 1과 같은 트리(Tree) 구조로서 원자(Atomic) 모델과 결합(Coupled) 모델로 계층화되어 있으며, 이에 대한 수학적 명세는 아래와 같다.

Atomic Model=<X, Y ,S, δext, δint, λ, ta>
X: 입력사건의 집합
Y: 출력사건의 집합
S: 모델 내부 상태의 집합
δext: $Q \times X \rightarrow S$: 외부 상태 천이 함수
$Q = \{(s,e) | s \in S, 0 \leq e \leq ta(s)\}$
s: 현재상태, e: 현재상태에 머문 시간
Q: 원자모델의 전체상태

λ: S → Y: 출력 함수
δint: S → S : 내부 상태 천이 함수
ta: S → R0,∞: 시간 진행 함수

Coupled Model=<X, Y, M, EIC, EOC, IC, SELECT>
X: 입력사건의 집합
Y: 출력사건의 집합
M: 하위 모델의 집합
EIC: External Input Coupling Relation
(외부 입력 사건 연결 관계)
EOC: External Output Coupling Relation
(외부 출력 사건 연결 관계)
IC: Internal Coupling Relation
(내부사건 연결 관계)
SELECT: 외부 입력 이벤트 처리에 대한 내부 모델 간의 우선 순위 선택 함수

이상의 2가지 모델은 모든 시뮬레이션 프로그램에서 반드시 기술하여야하는 시뮬레이션 모델의 구조와 동작을 설명하고 있다. 먼저 결합 모델은 DEVS 이론에서 모델 구조를 표현한다. 다시 말해 결합모델은 DEVS 이론에서 부체계(Subsystem Model)의 조립으로 나타나는 체계(System Model) 모델을 의미하는 것으로, 체계 모델 곧 상위 모델은 시스템 이론에 기초하여 여러 부체계의 조합으로 구성되어짐을 나타낸다. 그림 2에서 보듯이 하나의 체계는 DEVS에서 결합 모델로 나타난다. 그림은 하나의 체계가 계층적(Hierarchically)으로 구성되는 여러 부체계의 집합임을 보여주며, 부체계 간의 연결구조(Coupling)에 의해 체계 모델의 데이터 구조가 결정됨을 나타낸다.

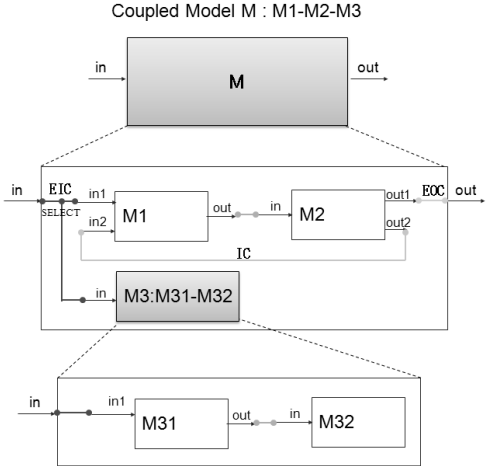


그림 2. 결합모델로 표현되는 DEVS 모델의 계층적 구조

다음으로 원자 모델은 DEVS 이론에서 모델의 동작을 나타낸다. DEVS 이론에서 모델의 동작은 오직 원자모델을 통해서만 기술된다. 그림 3은 원자모델의 동작을 설명하기 위해 나타낸 시간개념이 추가된 상태천이도(Timed Finite State Machine)로서 원자모델의 두가지 동작 원인을 설명하고 있다. 그림에서 보듯이 DEVS에서는 모델 동작의 원인을 실선으로 표현되는 외부이벤트(External Event)와 점선으로 표현되는 시간이벤트(Timed Event)로 정의한다. 그림에서 원자모델은 S1, S2의 두가지 상태를 가지고 있으며, 초기상태는 S1에서 출발한다. DEVS에서는 하나의 상태에 머물수 있는 시간(resident time)이 존재하며 이는 반원의 하단부에 시간(ta:time advance) 값으로 표시된다. 원자모델은 외부이벤트를 받으면 외부상태천이(External Transition) 함수를 호출하여 동작을 정의하며, 하나의 상태에서 정의된 resident time이 초과하면 내부상태천이(Internal Transition) 함수를 호출하여 이에 따른 동작을 정의한다. 아래 그림에서 외부 이벤트 수신시에는 외부상태천이함수에서 모델의 상태를 S1으로부터 S2로 바꾸도록 정의하고 있으며, S2에서 경과시간 ta를 초과하면 출력이벤트 detect를 발생시키고 모델의 상태를 S2에서 S1으로 바꾸도록 동작을 정의한다.

DEVS 형식론에서 모델은 외부에서 입력된 이벤트와 더불어 내부적으로 스케쥴링된 동작 시간에 맞춰 실행된다. 즉, DEVS의 시뮬레이션 알고리즘은 외부 입력 이벤트에 따라 모델의 외부 상태 천이 함수를 호출하고 내부 스케쥴링 이벤트(시간이벤트)에 따라 모델의 출력함수와 내부 상태 천이 함수를 호출토록 하는 것이다. 앞서 그림 1에서처럼 DEVS는 이러한 시뮬레이션 알고리즘을 처리하는 프로세서(processor)두고 이를 구성모델에 1:1로 대응시킨다. 시뮬레이션 프로세서는 원자모델에 대해서는

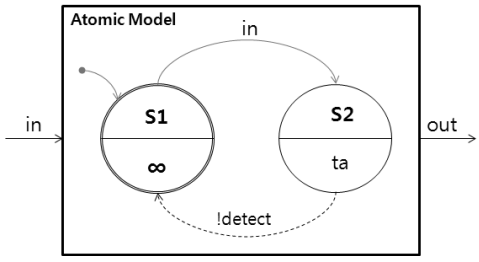


그림 3. 원자모델로 표현되는 DEVS 모델의 동작원리

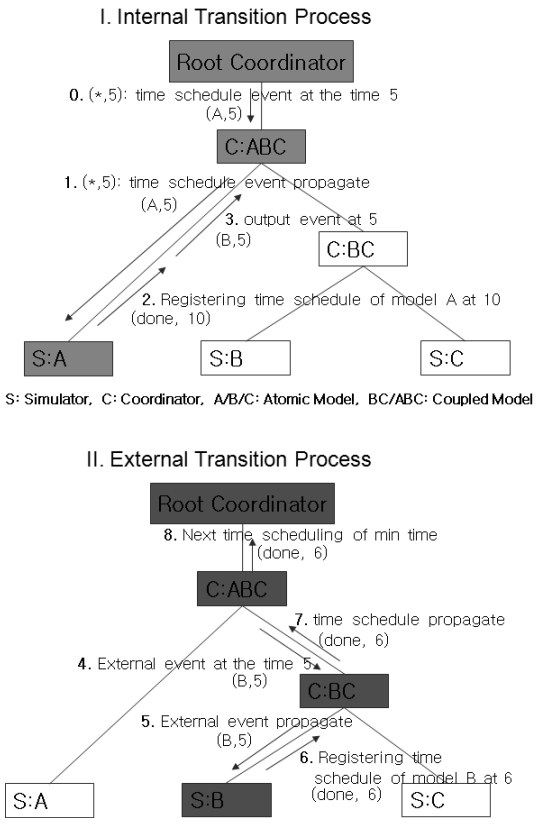


그림 4. DEVS 시뮬레이션 알고리즘

시뮬레이터(Simulator), 결합 모델에 대해서 코디네이터라 명칭하며, 이들은 기술한 대로 동작 시간에 맞춰 모델의 함수를 호출하고, 모델의 다음 동작 시간을 설정하여 상위 프로세서로 전파하는 역할을 수행한다. 최상위 코디네이터인 루트 코디네이터는 하부 모델로부터 전파된 시간 스케쥴링을 취합하고 모델의 시간에 따른 취합된 모델들의 시간 스케쥴 중에서 최소값을 다음 스케쥴 시간으로 설정한다. 그림 4는 기술된 DEVS 시뮬레이션 알고리즘의 처리순서를 Internal Transition 및 External Transition의 함수 호출에 맞춰 보여준다.

3. 매트랩 기반 DEVS 프레임워크

기술된 DEVS 형식론은 이산사건시스템의 시뮬레이션을 위한 명확하고도 간결한 개발 방법론을 제공하고 있기 때문에 C++, JAVA, Python, C#, XML과 같은 다양한 프로그래밍 환경에서 DEVS 형식론을 적용하여 시뮬레이션 응용 프로그램을 개발할 수 있는 DEVS 프레임워크가 개발되었다[11,12,13,14]. 본 논문은 이러한 기존의 DEVS 프레임워크 구축 연구에 대한 연장선상으로서 공학용 프로그래밍 언어로 광범위하게 사용되고 있는 매트랩으로 구현된 DEVS 프레임워크를 소개하고 있다. 본 연구에서 제안된 DEVS 프레임워크는 기존의 프레임워크와 비교할 때, 언어적 차별성 외에 매트랩/시뮬링크 기반으로 개발된 컴포넌트 모델을 별도의 연동 기법없이 DEVS 모델로 표현되는 교전 이벤트 모델과 즉시 결합할 수 있는 장점이 있다. 이에따라 실제적인 컴포넌트 기반의 무기체계 교전 시뮬레이션 개발 환경을 매트랩/시뮬링크에서 구축할 수 있다. 기본적으로 제안된 프레임워크의 개발 배경은 기존에 지적된 매트랩/시뮬링크의 단점, 곧 연속시간 및 이산시간 시스템과 같은 컴포넌트 단위의 모델링에서는 유용하지만 교전 시뮬레이션과 같이 객체지향 기반의 상위 체계의 조립성이 요구되는 시뮬레이션 환경에서는 모델링의 복잡성 및 구현 난이도가 높아지는 단점을 보완하기 위한 것이었다. 이를 위해 저자는 매트랩/시뮬링크 환경에 객체지향 기반의 이산사건시뮬레이션 형식론을 적용하여 무기체계 교전 시뮬레이션에서 요구되는 컴포넌트 조립을 통한 상위체계 모델의 조립 기능을 손쉽게 구현할 수 있도록 하였다. 이에 대한 예제로서, 그림 5는 기뢰지대를 통과하는 잠수함의 운용 시뮬레이션 프로그램을 개발하기 위해 시뮬링크로 기개발된 수중운동체의 운동 모델을 재사용하는 것을 보여주고 있다. 그림 5에서 보는 바와 같이 개발된 프레임워크에서는 DEVS 형식론을 기반으로 무기 체계에서 요구되는 체계/부체계 구조의 계층화된 교전 시뮬레이션 모델을 개발할 수 있을 뿐만 아니라, 모델의 개발과정에서 기동부/탐지부/제어부/추진부와 같이 매트랩/시뮬링크에서 기개발된 컴포넌트 모델을 손쉽게 재사용할 수 있는 장점을 제공한다.

DEVS 형식론은 객체 지향적 사고를 근간으로 하기 있기 때문에 절차지향프로그래밍의 특성을 지닌 2007년 버전까지의 매트랩에서는 사실상 적용이 어려웠다. 그러나 이후 버전에서는 객체지향 프로그램밍이 지원됨으로써 매트랩 기반의 DEVS 프레임워크가 개발될 가능성을 열어놓았다. 이를 계기로 저자는 2008년 3월부터 새로운 매트랩 기반 프레임워크의 개념과 개발 방향(Design Drivers)을 정립하고 구현 및 테스트를 수행하였다. 새로운 DEVS 프레임워크의 개발 과정에서는 DEVS 형식론의 구조와 기존에 개발된 프레임워크의 장단점들을 면밀히 분석하였으며, 더불어 DEVS 프레임워크의 구현에 활용할 수 있는 여러 객체지향 디자인패턴을 연구하였다[15]. 이에 따라 표 1은 개발과정에서 핵심적으로 참조한 기존의 DEVS

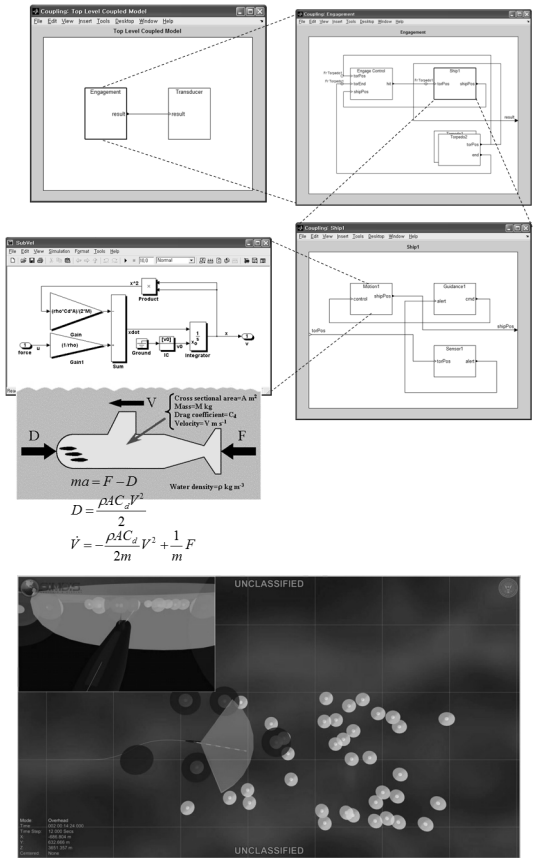


그림 5. 개발된 프레임워크를 활용한 컴포넌트 조립환경 (잠수함 기뢰회피 시뮬레이션)

표 1. 매트랩 기반 프레임워크의 개발방향

프레임워크	장점
DEVSim++	직관적이고 단순한 DEVS 함수구성 및 강력한 시뮬레이션 통제 및 분석 기능
DEVSJAVA	모델의 시각적 전시 및 구성

형식론 구현	활용 가능한 디자인패턴
계층적 모델구성	컴포지트 패턴
결합모델 연결	옵져버/커맨드/스트레티지 패턴
원자모델 실행	스테이트 패턴

프레임워크에 대한 분석의 내용과 개발 프레임워크에 적용한 소프트웨어 디자인 패턴을 나타내고 있다.

새로운 프레임워크의 개발 방향 중에서 핵심은 프레임워크의 단순화(Simplicity)였다. 기존 프레임워크의 구조 분석 및 적용 가능한 디자인패턴에 대한 분석 결과, 결합모델 구조를 컴포지트 패턴으로 구현함으로써 기존 프레임워크에서 Coupled 클래스를 생략할 수 있게 되었으며

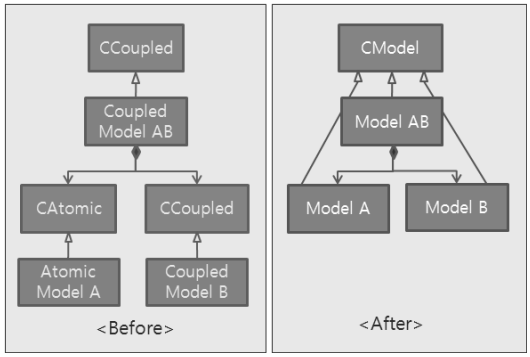


그림 6. 개발된 프레임워크의 모델 계층도

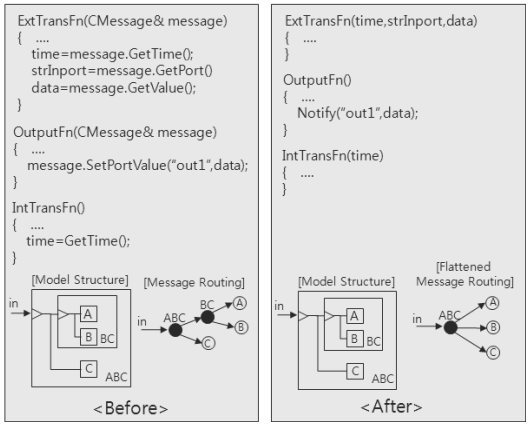


그림 7. 개발된 프레임워크의 메시지 전달 방식

(그림 6), 기존 프레임워크에서 Message 클래스를 통한 이벤트 전달 및 시뮬레이션 정보 획득 구조를 대체하여 모델 함수의 호출 시에 이벤트정보를 함수의 입력 매개변수로 직접 전달하였다. 더불어 계층적 모델 구조로 인해 메시지 트리(Tree)로 전달되던 기존 메시지 전달방식을 단순화 하여, 내부적으로 메시지 트리를 없애는 직접전달 기법(Flattening)을 적용하였다(그림 7). 더불어, 기존의 시뮬레이션 알고리즘에서 원자모델 및 결합모델에 대한 1:1로 각각의 시뮬레이터들이 존재했던 것을 전체 모델에 대해 오직 하나의 시뮬레이터 곧, 루트 코디네이터만 존재토록 하였다. 대신 외부입력 이벤트의 전파는 각각의 모델이 연결 구조를 바탕으로 자체적으로 수행토록 하였으며, 루트 코디네이터에서 모든 원자모델에 대한 시간 관리만을 전담토록 함으로써 기존의 이벤트 스케쥴링 기법과 동일한 결과를 얻도록 하였다(그림 8). 그러나 이러한 단순화는 단지 프레임워크의 내부적 변화이며 프레임워크의 클래스 API, 문법 및 활용 방식은 기존(DEVSim++)

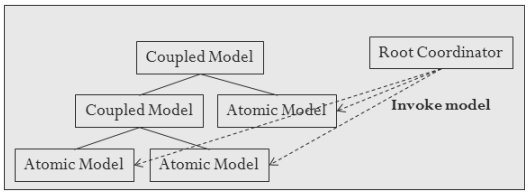


그림 8. 개발된 프레임워크의 시뮬레이션 알고리즘

과 유사하게 하였다. 결과적으로 새로운 프레임워크에서는 기존과 마찬가지로 DEVS 형식론의 기능을 충실히 반영하면서도 실제적으로 고려해야 하는 베이스클래스가 단지 2개의 클래스-Simulator와 Model 클래스-에 지나지 않을 정도로 단순화되었다. (추가적으로 모델 가시화를 위한 Model Viewer 클래스가 매트랩 DEVS 프레임워크에서 핵심적으로 제공된다.)

그림 7의 함수명은 모델 작성을 위해 모델 클래스에서 제공되는 기본 인터페이스를 나타내고 있다. 그림 7의 왼쪽이 개발 프레임워크의 비교대상인 DEVSim++의 원자모델 클래스를, 오른쪽이 개발된 프레임워크의 원자모델 클래스 인터페이스를 보여주고 있다.

이상의 단순하고 새로운 구조를 덧붙여 새롭게 개발된 프레임워크는 강력한 시뮬레이션 모델 가시화 및 분석 기능과 병렬 컴퓨팅을 이용한 향상된 연산 능력을 제공한다. 개발된 프레임워크가 제공하는 모델 구조의 가시화 기능은 구현된 응용 프로그램으로부터 모델 구조를 추출하게 된다. 따라서 구현 단계의 산출물 결과를 설계 단계와 비교 할 수 있는 수단을 제공한다. 더불어 시뮬레이션 진행 과정동안 생성된 객체들의 이벤트타이밍과 메세지의 전파과정을 가시화함으로써 모델의 호출 순서와 실행 과정을 직관적으로 파악할 수 있도록 한다(그림 9,10,11).

기술된 모델 분석 기능에 덧붙여 개발된 프레임워크는 시뮬레이션 로깅을 통한 사후분석(post simulation analysis) 기능을 제공한다.(그림 12) 새로운 프레임워크의 로깅 기능은 기존의 ASCII 파일 로깅 방식과 달리 매트랩의 행렬 타입으로 로그 파일을 작성하게 된다. 따라서 매트랩의 메모리 공간에 로그를 기록 분석하기 때문에 기존의 ASCII 파일 로깅 방식에 비해 성능 저하를 최소화 할 수 있다. 더불어 보다 효과적인 로그 분석과 모델 개발 과정에서의 구현 상 문제점을 검사(inspection)하기 위해 개발된 프레임워크에서는 로그를 차트로 전시하여 시뮬레이션 시간 대비 프로세싱시간을 직관적으로 파악하여 구현된 함수의 성능에 대한 프로파일링(profiling)이 가능토록 하였다.(그림 13)

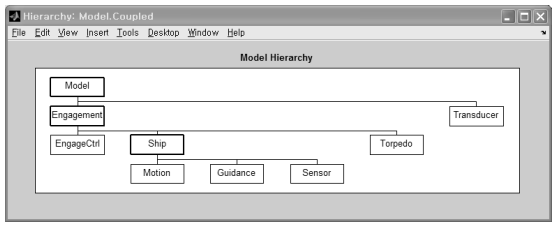


그림 9. 모델의 구조 가시화 기능

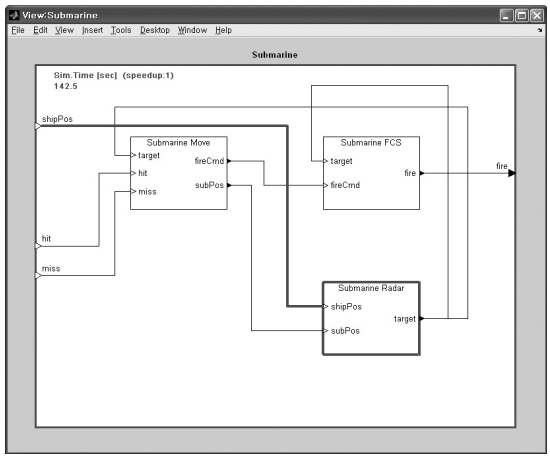


그림 10. 모델의 연결 관계 및 실행 가시화 기능

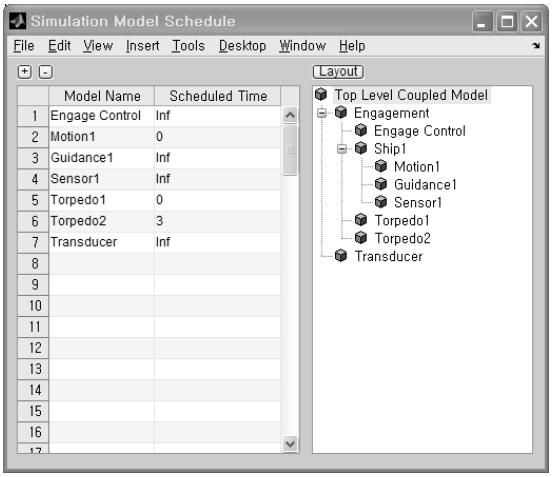
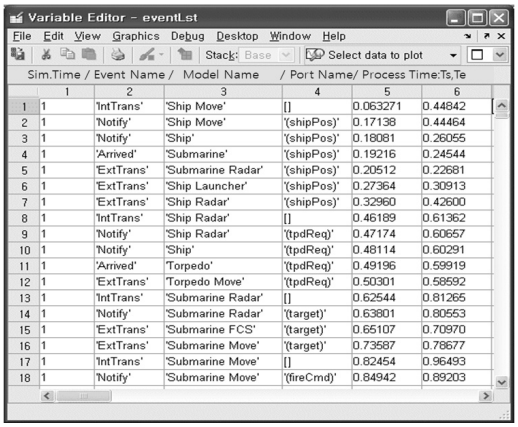


그림 11. 모델 객체의 런타임 이벤트 스케쥴 가시화

끝으로 개발된 프레임워크의 시뮬레이션 배치(Batch) 실행 기능은 몬테카를로 시뮬레이션과 같은 반복 시뮬레이션 환경에서 시뮬레이션 모델의 변수 설정과 반복 실행을 간단하게 수행토록 지원한다. 일반적으로 몬테카를로 시뮬레이션에서는 수백 회에서 수천 회에 이르는 반복 시



	1	2	3	4	5	6
1	1	'IntTrans'	'Ship Move'	[]	0.063271	0.44842
2	1	'Notify'	'Ship Move'	'(shipPos)'	0.17138	0.44464
3	1	'Notify'	'Ship'	'(shipPos)'	0.18081	0.26055
4	1	'Arrived'	'Submarine'	'(shipPos)'	0.19216	0.24544
5	1	'ExtTrans'	'Submarine Radar'	'(shipPos)'	0.20512	0.22681
6	1	'ExtTrans'	'Ship Launcher'	'(shipPos)'	0.27364	0.30913
7	1	'ExtTrans'	'Ship Radar'	'(shipPos)'	0.32960	0.42600
8	1	'IntTrans'	'Ship Radar'	[]	0.46189	0.61362
9	1	'Notify'	'Ship Radar'	'(tpdReq)'	0.47174	0.60657
10	1	'Notify'	'Ship'	'(tpdReq)'	0.48114	0.60291
11	1	'Arrived'	'Torpedo'	'(tpdReq)'	0.49196	0.59919
12	1	'ExtTrans'	'Torpedo Move'	'(tpdReq)'	0.50301	0.58592
13	1	'IntTrans'	'Submarine Radar'	[]	0.62544	0.81265
14	1	'Notify'	'Submarine Radar'	'(target)'	0.63801	0.80553
15	1	'ExtTrans'	'Submarine FCS'	'(target)'	0.65107	0.70970
16	1	'ExtTrans'	'Submarine Move'	'(target)'	0.73587	0.78677
17	1	'IntTrans'	'Submarine Move'	[]	0.82454	0.96493
18	1	'Notify'	'Submarine Move'	'(fireCmd)'	0.84942	0.89203

그림 12. 시뮬레이션 로그 분석 기능

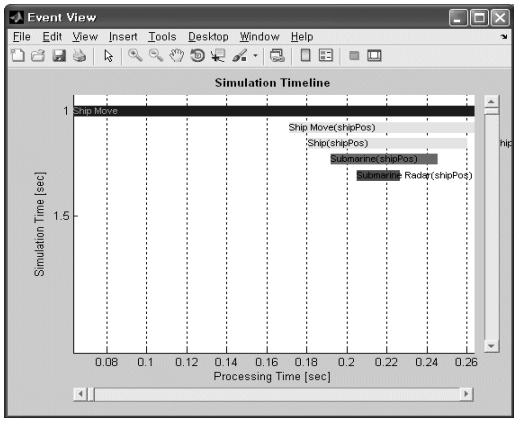


그림 13. 시뮬레이션 로그 가시화 기능

뮬레이션 동안 주요 인자(factor)를 가변 시키면서 변인의 영향성을 분석하게 된다. 이러한 상황에서 그림 14와 같은 배치기능은 보다 효과적인 반복 시뮬레이션을 가능케 한다.

더욱이 배치기능은 매트랩의 병렬 컴퓨팅 기술을 기반으로, 그림 14에 나타난 것처럼 반복 시뮬레이션 과정에서 멀티코어 CPU 코어의 수에 비례하여 가속화된 연산 기능을 제공한다. 일반적으로 병렬 컴퓨팅을 수행하기 위해서는 컴퓨팅 환경에 맡게 추가적인 병렬 컴퓨팅 관련 프로그래밍을 수행하여야 하지만, 그림 14에서처럼 개발된 프레임워크에서는 병렬 컴퓨팅 기능이 자동적으로 지원됨으로 개발자가 별도의 프로그래밍 작업을 할 필요가 없다. 그림 15는 병렬 컴퓨팅을 통한 성능향상을 시험한 결과로서, 동일한 어뢰방어(Torpedo Defense) 시뮬레이션 응용 프로그램을 언어(C++/Matlab)의 문법만 바꿔서

```
% 단일 시뮬레이션 인스턴스 생성 및 실행
rSim=Simulation.TorpedoDefense;
rSim.Start;

% 배치 시뮬레이션 인스턴스 생성 및 실행
rSimArray(80,4)=Simulation.TorpedoDefense;
rSimArray.Batch(runs);  %runs: 배치파라메터
%-------------------------------------------------------
% ∵ runs(1)                runs(2)      ....
%   LmodelName='A';        LmodelName='A';
%   LpropertyName='speed'  LpropertyName='speed'
%   Lvalue=100;            Lvalue=200;
%
% :배치 시뮬레이션 인스턴스 생성/배치실행
%   →4회 run에 각각 80회의 trial 생성 (320회 반복실행)
%-------------------------------------------------------

rSimArray.Start;
```

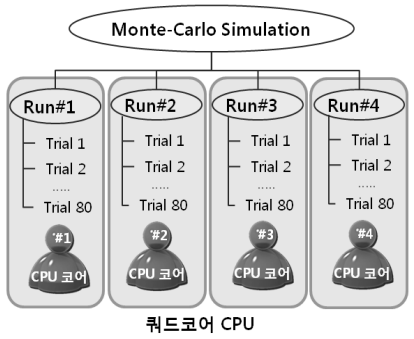
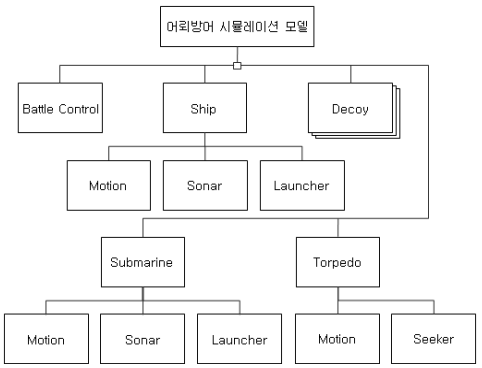


그림 14. 개발된 프레임워크의 배치실행 기능



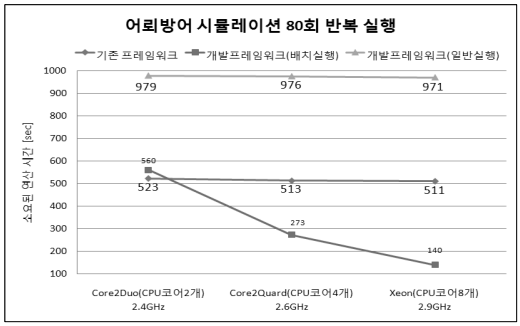


그림 15. 개발된 프레임워크의 병렬 컴퓨팅 성능

DEVSim++와 개발된 프레임워크에서 80회 반복 시뮬레이션 했을 때 소요된 연산 시간이다.

4. 결 론

본 논문은 객체지향 컴퓨터 시뮬레이션 프로그램의 개발 프레임워크로 새롭게 개발된 매트랩 이산사건 시뮬레이션 프레임워크의 개발 과정 및 주요 기능을 소개하고 있다. 새롭게 개발된 프레임워크는 이산사건시뮬레이션 형식론(DEVS Formalism)의 뛰어난 개발 방법론과 매트랩 언어의 편리성을 접목시키고 있다. 더불어 기존 C++/JAVA 환경에서 개발된 DEVS 프레임워크의 장점을 취합하고 , 컴퓨팅 성능을 향상시킬 수 있는 다양한 기법을 적용함으로써 보다 단순한 구조와 향상된 성능을 가진 새로운 프레임워크가 되었다. 이외에도 지면관계로 상세히 소개하지 못했지만, 개발 프레임워크는 매트랩 기반으로 이산 사건을 모델링하기 때문에 미분 및 차분방정식으로 표현되는 연속사건을 모델링하는데 뛰어난 시뮬링크와 별도의 연동어댑터없이 직접연동을 수행할 수 있다. 따라서 손쉬운 하이브리드(연속+이산사건) 시뮬레이션 환경을 구축할 수 있다.

본 논문에 기술된 개발 프레임워크는 다양한 범용 시뮬레이션 프로그램 개발을 지원하며, 특히, 국방 분야의 무기체계 시뮬레이션과 같이 부체계의 조립으로 상위체계가 구성되는 시스템 레벨의 객체지향 시뮬레이션 환경에서 보다 효과적으로 활용 될 수 있다.

참 고 문 헌

1. J. T. Buck, S. Ha, E. A. Lee, D. G. Messerchmitt, “Ptolemy: A Framework for Simulating and Prototyping Heterogeneous Systems”, Int. Journal of Computer Simulation, Vol. 4, pp. 155-182, 1994.
2. Herb Schwetman, “User’s Guide: CSIM18 – The Simulation Engine”, Mesquite Software.
3. Felix Breitenecker, “Software for Modeling and Simulation – History, Developments, Trends and Challenges”, Simulation and Visualization 2006.
4. Guido Sandmann, “The MathWorks at a Glance”, MATLAB & SIMULINK Brochure 2008.
5. F.Bouchhima et al., “A SystemC/Simuink Co-Simulation Framework for Continuous/Discrete Events Simulation”, IEEE Rapid System Prototyping Proceedings, 2005.
6. Chang Ho Sung et al., “Interoperation of DEVS Models and Differential Equation Models using HLA/RTI”, Proceedings of the 2009 Spring Simulation Multiconference, 2009.
7. Peter Fritzson, “Principles of Object-Oriented Modeling and Simulation with Modelica 2.1”, IEEE Press, 2004.
8. Bernard P. Zeigler, et al. “Theory of Modeling and Simulation”, ACADEMIC PRESS, 2001.
9. Tag Gon Kim, “DEVSim++ ver3.0 Developer’s Manual”, http://smslab.kaist.ac.kr, 2006.
10. Bernard P. Zeigler, “DEVSJAVA ver3.0 User Reference Guide”, Arizona Univ. ACIM, 2003.
11. Moon Ho Hwang, “DEVS#: C# Open Source Libray of DEVS Formalsim”, http://xsy-csharp.sourceforge.net/DESV sharp, 2007.
12. J. L. Risco-Martin, S. Mittal, M. A. Lopez-Pena, J. M. de la Cruz, “A W3C XML Schema for DEVS Scenario”, Proc. of the 2007 Spring Simulation Multiconference, pp 27-286, 2007.
13. Jean-Sebastien Bolduc, Hans Vangheluwe, “A Modeling Simulation Package for Classic Hierachical DEVS”, http://msdl.cs.mcgill,ca/projects/DEVS/PythonDEVS, 2002.
14. V. Janousek, and E. Kironsky, “Exploratory Modeling With SmallDEVS“, Proc. of the 20th Annual European Simulation and Modeling Conference, 2006.
15. Andrey V. Pristupa, “Design Patterns In Discrete Event Simulation”, IEEE KORUS proceedings, 2004.

황 근 철 (hkchul@add.re.kr)

2001 경북대학교 전자전기공학부 학사
2003 서울대학교 전기컴퓨터공학부 석사
2003～현재 국방과학연구소 제6기술연구본부 연구원

관심분야 : 무기체계 모델링&시뮬레이션, 체계시뮬레이션 및 체계성능분석 (System Simulation & System Operational Performance Analysis), 모델기반 시뮬레이션, 시뮬레이션 프레임워크

이 민 규 (messin@add.re.kr)

2006 경북대학교 전자전기공학부 학사
2008 한국과학기술원 전기전자공학부 석사
2008～현재 국방과학연구소 제6기술연구본부 연구원

관심분야 : 모델링&시뮬레이션, 모델기반 시뮬레이션, 전투체계 시뮬레이션, 무기체계 효과도분석, 무기체계 전술분석

김 정 훈 (kimjh@add.re.kr)

1998 서울대학교 해양학과 학사
2001 서울대학교 지구환경과학부 석사
2001～현재 국방과학연구소 제6기술연구본부 선임연구원

관심분야 : 국방 모델링&시뮬레이션(Defense Modeling and Simulation), 무기체계 효과도 분석(Weapon Systems Effectiveness Analysis), 해상/수중무기체계 전투실험(Battle Experiments for Naval Weapon System)