SBA를 위한 M&S PlugIn-Based Architecture(PBA) 구조에 대한 연구

Study on M&S PlugIn-Based Architecture(PBA) for SBA

원 강 연* 최 상 영**
Won, Garng-Yun Choi, Sang-Yeong

ABSTRACT

Simulation Based Acquisition(SBA) gains interest from the defense acquisition community. To accomplish SBA efficiently, M&S should be collaborately used in. This paper proposes PBA(PlugIn-Based Architecture) that is a common software development infrastructure and provides reuse mechanism with plugin components. PlugIns are reuse entity possible to plug-in-play statically and dynamically. This architecture supports stand alone simulation and HLA-compliant distributed simulation also.

주요기술용어(주제어) : Modeling(모델링), Simulation(시뮬레이션), Reuse(재사용), Architecture(아키텍처), Component (구성품), Plug-and-Play(플러그엔 플레이), Middleware(미들웨어), Framework(프레임워크), Plugin(플러그인)

1. 머리말

무기체계 소요분석, 개발, 시험평가, 훈련 등의 획득에 있어 모델링 및 시뮬레이션(M&S)의 활용이 중요시되고 그 비중 또한 점차 늘어나고 있으나 업무 적용 면에서는 획득 상의 단위 업무별로 산별적인 개발과 활용에 그치고 있는 실정이다.

이러한 무기체계 획득 상의 비효율적인 M&S 업무 관행을 개선하기 위하여 Simulation Based Acquisition (SBA)의 정착이 요구된다. SBA는 무기체계 소요분석 및 결정에서 분석/설계, 제작, 시험/평가, 훈련/운용/군수지원에 이르는 전 과정 상에서 각 단계에 대해 상호 통합적 협력으로 M&S를 활용함으로써 획득 전 과정 상의 시간, 자원, 위험을 실질적으로 감소시킬 수 있으며 운영유지 비용을 줄이고 배치된 체계의 품질, 운용성 등에 대한 효과도 증대를 지향한다[1].

SBA를 효과적으로 수행하기 위해서는 각 획득 사업 담당자 간, 획득 단계 간, 더 나아가서는 성능개선, 차기사업 그리고 타 사업 간의 협업이 중요하다. 이를 위해서는 공학설계 담당자 간의 모델과 데이터의 공유뿐만 아니라 소요분석, 개념연구, 주요 요구사항 설정, 효과도 분석 등에 있어서 획득 담당자들과 제작, 개발/운용 시험평가, 교육훈련 등의 업무에 재

† 2006년 9월 7일 접수~2007년 1월 12일 게재승인
* 국방과학연구소(ADD)
** 국방대학교 국방과학처(National Defense University)
주저자 이메일 : wongyn@empal.com

사용이 필요하다.

이러한 협업은 사업초기부터 각 획득 담당자들의 지속적인 사업 참여로 피드백을 통한 각 기능의 연계를 용이하게 하고 반복적인 개발루프와 동시공학적 획득 프로세스를 달성할 수 있을 것이다.

획득 프로세스에 있어 M&S의 적용을 통한 협업을 촉진하기 위해서는 위에서 언급한 협업 문화와 획득 프로세스 정립과 더불어 이를 효율적으로 수행할 수 있도록 하부 기반환경의 구축이 필요하다. 기반환경은 분산 개발환경에서 자료의 흐름을 원활하게 해주며 재사용성 및 상호운용성을 높일 수 있는 기반체계를 지향한다.

이러한 노력으로 미군은 High Level Architecture (HLA)[2], Joint Modeling and Simulation System (JMASS)[3] 등의 시뮬레이션 시스템 아키텍처를 개발하였다. HLA는 독립적으로 개발된 시뮬레이터간의 상호운용성을 보장하기위한 분산 시뮬레이션 아키텍처로서 2000년에 IEEE 1516 표준으로 자리매김하고 있으며 JMASS는 1998년 JMASS Joint Program Office(JPO)에 의해 개발을 시작하여 지속적인 성능 개선 및 적용을 추진하고 있다.

JMASS는 시뮬레이션 엔진 및 관련 서비스, 인터페이스 규칙, 제반 도구 등으로 구성된 아키텍처 프레임워크이다. 모델의 구성요소는 최상위 모델 컴포넌트인 Player와 Player를 구성하는 Component들로 이루어진다. 이들 Player간의 통신은 Ports를 통해 이루어지며 Player들이 결합되어 Team이 형성되고 실행이 가능한 형태가 된다.

이렇게 공통 규격에 맞게 생성된 JMASS의 Players는 목적에 따라 특정 Team으로 구성되어 시뮬레이션 될 수 있는 재사용 가능한 M&S 구성 요소를 지향한다[4].

하지만, JMASS의 재사용 개체인 Players와 Players간의 자료교환을 위한 각각의 Ports와 커플링(Coupling)의 생성은 코드 생성 시 이루어지므로 실행 간 시뮬레이션 개체의 동적인 생성 및 재구성이 불가능하며 Player에 의해 특정 객체를 구현하는 경우 실행 시 해당 객체에 대해 다수의 인스턴스(Instance)를 생성할 수 없는 구조를 가지고 있다[5].

시뮬레이터 간 연합 시뮬레이션을 위한 HLA는 분산연동 기반서비스를 통해 시뮬레이터 단위의 재사용성 및 확장성을 높일 수 있는 아키텍처이나 단일 시뮬레이터 내의 컴포넌트에 대한 재사용성 및 확장성을 제공하지는 않으며 단일 시뮬레이터의 아키텍처와 시뮬레이션 엔진 등의 기반 서비스를 제공하지는 않는다.

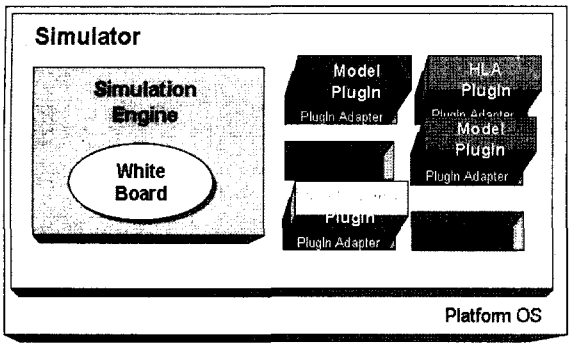
본고는 이와 같이 기존 M&S 아키텍처가 갖는 제약사항을 고려하여 시뮬레이션 컴포넌트 간의 커플링(Coupling)이 없이 상호 독립적인 인터페이스 및 상호작용을 지향함으로써 컴포넌트가 쉽게 추가되고 조합될 수 있으며 시나리오에 따라 plug-and-play 방식으로 재구성이 용이하고 시뮬레이션 실행 시에도 구성품의 생성/삭제(hot-swap)가 가능한 M&S PlugIn -Based Architecture(PBA)를 제안하고자 한다.

본고의 구성은 제2장에서 하부기반으로 제공되는 시뮬레이션 엔진과 PlugIn들로 구성되는 PBA의 구조에 대해 설명하고 제3장은 모델을 구현하며 재사용 가능한 컴포넌트인 PlugIn의 내부 구조와 시뮬레이션 엔진과의 연동 방식을 기술한다. 제4장에서는 PlugIn 간의 연동방식에 대하여 설명하고, 제5장은 PlugIn들에 대한 스케줄링 및 동기화에 대해 설명한다. 제6장에서는 PBA의 HLA 분산 시뮬레이션 적용 구조에 대하여 설명하고, 제7장에서 결론을 맺는다.

2. PBA 구조

PBA는 이산사건 시스템(Discrete Event System)을 월드뷰(World View)로 하며 그림 1과 같이 사건 관리와 스케줄링 및 동기화를 담당하는 시뮬레이션 엔진과 시뮬레이션 S/W를 구성하는 빌딩블록인 PlugIn으로 이루어진다. 그리고 S/W 컴포넌트인 PlugIn 간의 커플링(Coupling)을 최소함으로써 상호 독립성을 보장하기 위해 시뮬레이션 엔진을 통해 메시지 교환을 수행하는 Independent Component Architecture[6] 스타일을 갖는다.

이렇게 상호 독립성을 갖는 PlugIn의 특성은 수정을 용이하게 할 뿐만 아니라 정적 혹은 동적으로 plug-and-play 방식의 시뮬레이션 구성과 독립적인 PlugIn 구현이 가능함으로써 재사용성을 높일 수 있



[그림 1] PBA 구조

도록 해준다.

PlugIn은 시뮬레이션 하고자하는 개체 단위 뿐만 아니라 대상 개체에 대한 구성품 단위로 대응될 수 있으며 지형, 대기 등과 같은 공통적인 요소 등에 대한 모델을 단위 PlugIn으로 구현할 수 있다. 따라서 대상 체계의 구성품뿐만 아니라 전장 및 교전 환경에 따라 시뮬레이션 요소 구현 및 구성이 용이하며 유연하게 대처할 수 있다.

특히, 시나리오에 따른 시뮬레이션 초기화 단계뿐만 아니라 시뮬레이션 수행 시에도 동적인 PlugIn 생성이 가능하므로 표적 등과 같이 다수의 동종 혹은 이질적인 시뮬레이션 개체가 필요한 경우, 해당 PlugIn을 추가적으로 다수 생성하거나 삭제가 가능하므로 기존에 동일 모델의 개체를 다수 생성하는 경우 리스트 형태로 관리해야 하는 난점을 해결할 수 있다.

PlugIn은 아키텍처 프레임워크로 제공되는 PlugIn Adapter를 통해 시뮬레이션 프레임에 접속된다. PlugIn Adapter는 PlugIn이 공통 인터페이스 통해 시뮬레이터 구성에 참여할 수 있도록 한다. 따라서 각 PlugIn은 이러한 인터페이스 규칙과 공통 기반 서비스를 바탕으로 모델과 시뮬레이터를 분리함으로써 구현을 용이하게 할 수 있으며 독립적으로 기 개발된 PlugIn을 사용 목적에 따라 모델 충실도 수준(Fidelity Level)이나 일부분을 수정과 보완함으로써 재사용을 용이하게 할 수 있는 특징을 갖는다.

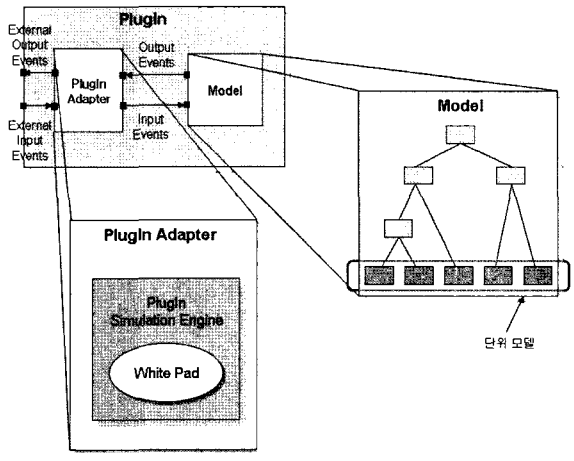
시뮬레이션 엔진의 White Board(WB)는 생성되거나 삭제되는 PlugIn의 관리를 담당하며 접속된 PlugIn과 이들 간의 자료 교환 정보에 대한 유지관리와 자료배포를 수행함으로써 Publish/Subscribe 방식의 사건중심 PlugIn 연동 방식을 제공한다. 시뮬레이션 엔진은 이러한 WB의 자료 배포 서비스와 각 PlugIn의 시뮬레이션 진행 시각 정보를 바탕으로 PlugIn 스케줄링과 시뮬레이션 동기화를 수행한다. 특히, 실시간 제약조건 하에서는 Real Time Clock(RTC)을 바탕으로 실시간 시뮬레이션 스케줄링을 제공한다.

3. PlugIn 구조

PlugIn은 시뮬레이션 구성 요소에 대한 모델을 구현하며 동적으로 생성되어 'Plug-In' 되고 상호 연결되어 작동될 수 있도록 하는 구조적인 공통 기반 인터페이스와 자체 시뮬레이션 엔진을 통한 독립적인 실행구조를 갖는다. 이러한 인터페이스와 지역 시뮬레이션 엔진은 그림 2에 나타낸바와 같이 아키텍처 프레임워크로 제공되는 PlugIn Adapter에 의해 수행된다.

PlugIn Adapter의 자체 시뮬레이션 엔진인 PlugIn Simulation Engine은 계층구조를 갖는 Coupled Model[7]의 모델 간 연결과 사건교환을 White Pad (WP)를 통해 구현한다.

PlugIn Adapter는 PlugIn 생성 시에 전역 시뮬레이션 엔진의 WB에 자신을 등록하고 정의된 단위모델들의 인스턴스를 생성한다. 이렇게 생성되는 각 단위모델은 모델 자신과 아울러 송수신하고자 하는 사



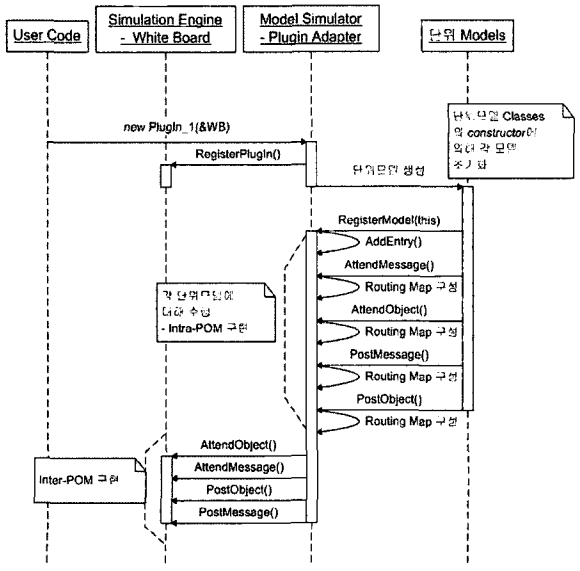
[그림 2] PlugIn 구조

건을 PlugIn Adapter의 WP에 등록한다. 이러한 각 단위모델에 의한 송수신 선언은 PlugIn 내부의 단위 모델 간 정보교환 모델인 Intra-POM(PlugIn Object Model)을 구현하게 된다.

WP는 등록된 송수신 사건 목록을 바탕으로 PlugIn 내부 모델 간 지역 사건 배포를 수행하며 PlugIn들 간의 정보교환 모델인 Inter-POM(PlugIn Object Model)을 바탕으로 전역 시뮬레이션 엔진의 WB에 PlugIn 외부와의 송수신 사건을 등록하고 시뮬레이션 수행 시 PlugIn 외부와의 사건 송수신을 대행한다.

이러한 PlugIn Adapter의 내/외부 사건의 중계 서비스는 해당 사건을 수신하는 단위모델에 대해 수신 사건의 time stamp로의 시간 전진과 작업을 배정한다. 그리고 WP 내의 모든 사건의 배포가 이루어진 경우는 각 단위 모델의 시간전진 요청 값 중 최소인 단위모델에 대해 해당 시각으로의 전진을 허용시킴으로써 시뮬레이션의 진행과 작업배정을 하는 등 PlugIn 자체에 대한 시뮬레이션 엔진 기능을 제공한다.

PlugIn의 모델부분은 단위모델들로 구성된다. 모델은 일반적으로 계층적인 구조의 하부모델로 표현될 수 있으나 모델의 구현 측면에서는 모델 계층의 단말 노드에 해당하는 단위모델들과 이들 간의 인터페이스만으로 표현이 가능하다. 이러한 모델의 구현 방식은



[그림 3] PlugIn 생성 및 초기화

설계된 모델의 계층구조를 구현에 반영하지 않음으로써 가독성이 떨어지는 단점을 갖고 있다. 하지만 PBA는 하부 모델 간에 전달되는 각 사건 별로 1:1 포트(Port)와 커플링을 생성하여 사건을 전달하는 구조를 지양하고 WP 서비스를 기반으로 송수신 선언을 통한 사건중심 메시지 중계를 지향한다. 이를 통해 유연하고 효율적인 모델 설계 및 시뮬레이션 구현이 가능하며 모델 간 커플링을 최소화함으로써 모델 갱신 및 수정에 따른 유지보수 노력을 줄일 수 있을 것이다.

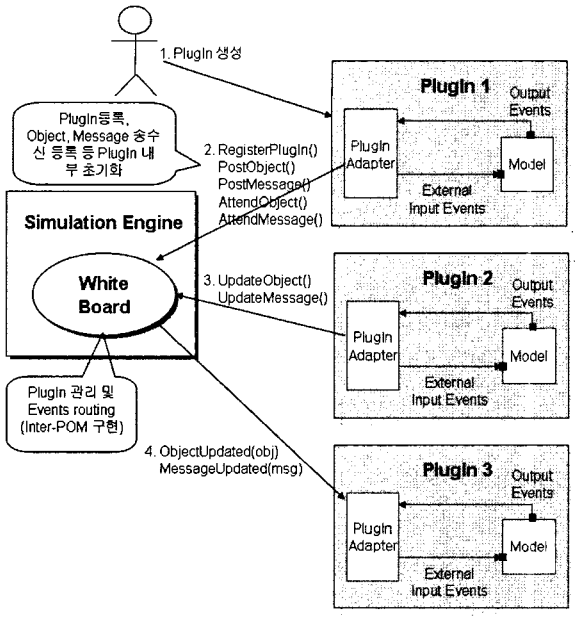
4. PlugIn 연동

PlugIn은 WB 서비스를 통하여 데이터를 주고받는다. 이를 위해 WB는 생성되는 PlugIn 정보를 등록하여 관리 유지하고 이들 간에 주고받는 데이터 종류와 송수신 맵을 구성한 후 store and forward 방식으로 수신되어 저장된 사건을 이를 수신하는 각 PlugIn에게 순차적으로 전달한다.

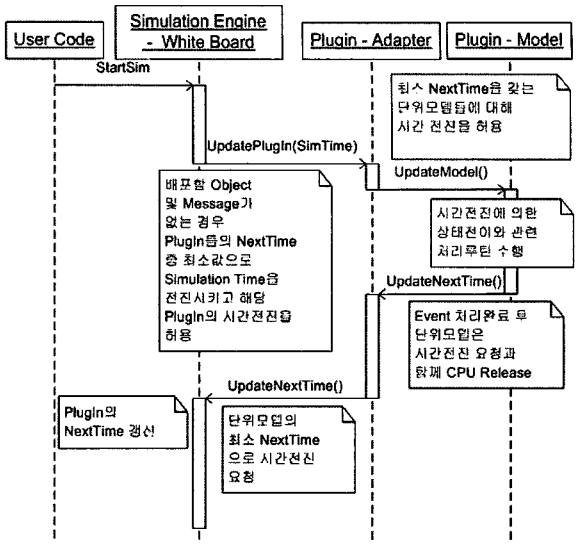
시나리오와 사건에 의해 생성되는 각각의 PlugIn은 그림 4에 나타낸 바와 같이 생성 시에 내부 모델의 초기화와 함께 자신과 자신이 배포할 데이터를 PostObject()와 PostMessage()를 통하여 WB에 등록하고 수신하고자 하는 데이터는 AttendObject()와 AttendMessage()를 통하여 WB에 등록한다. 이러한 PlugIn 간의 교환 데이터는 물론 사전에 Inter-POM (Inter-PlugIn Object Model)으로서 설계되어 정의된 사양을 따른다.

PlugIn은 배포 선언한 데이터의 갱신 시나 전송 주기에 따라 WB에 UpdateObject(), UpdateMessage()를 통하여 비동기식으로 WB에 전달하며 WB는 저장된 사건에 대하여 데이터 송수신 맵에 따라 해당 데이터에 대해 수신 선언한 PlugIn들에게 ObjectUpdated(), MessageUpdated()를 통하여 동기식으로 우선순위에 따라 전달한다.

PlugIn 간의 교환 데이터 유형은 Object와 Message로 구분된다. Object는 시뮬레이션 개체로서 다소의 영속성을 갖으며 자체의 속성을 지닌다. Message는 특정 종류의 사건정보를 담으며 Field들로 구성된다.



[그림 4] PlugIn 연동



[그림 5] 시간전진에 의한 스케줄링 및 동기화

5. PlugIn의 스케줄링 및 동기화

PBA는 하나 이상의 PlugIn들이 연합하여 시뮬레이션을 수행하는 구조이다. 따라서 시뮬레이션을 구성하는 이들 PlugIn들에 대한 작업시기 배정이 필요하며 동시에 시뮬레이션 시간 동기화가 이루어져야 한다. 이러한 PlugIn의 스케줄링과 동기화는 시간전진에 의한 방식과 사건에 의한 방식으로 이루어진다.

시간전진에 의한 방식은 그림 5에 나타낸 바와 같이 WB에 배포할 Object 및 Message가 없는 경우 시뮬레이션 엔진은 각 PlugIn들이 등록한 자신의 next scheduled time(이후 NextTime) 중 최소 NextTime으로 시뮬레이션 시각을 전진 시킨 후 해당 PlugIn에 대해 UpdatePlugIn을 통하여 갱신된 시뮬레이션 시각으로 전진을 허용한다.

UpdatePlugIn에 의해 전역 시뮬레이션 엔진으로부터 작업배정을 받은 PlugIn은 PlugIn Adapter에 의해 최소 NextTime 즉, 현재 시뮬레이션 시각을 갖는 단위모델에 대해 UpdateModel을 통해 시간전진을 허용시킴으로써 해당 단위모델의 상태전이와 관련 처리루틴을 수행토록 한다. 단위모델은 사건을 발생시키는 등 상태를 갱신시키고 시간전진이 필요하게 될 때 UpdateNextTime을 통해 자신의 NextTime을 PlugIn Adapter에게 알림으로써 CPU를 Release한다. 이때, 작업을 수행한 단위모델에 의해 출력된 사건이 PlugIn Adapter의 WP에 쌓여 있는 경우 Adapter는 단위모델 생성 시 메시지 송수신 선언으로 작성된 PlugIn 내부 이벤트 라우팅 테이블을 바탕으로 각 수신 단위모델에 대해 사건을 우선순위에 따라 순차적으로 전달함으로써 단위모델들에 대한 작업배정이 이루어지며 배정을 받은 단위 모델은 상태갱신을 통하여 모의를 진행하고 시간전진이 필요하게 될 때 UpdateNextTime을 호출함으로써 자신의 작업을 중단한다.

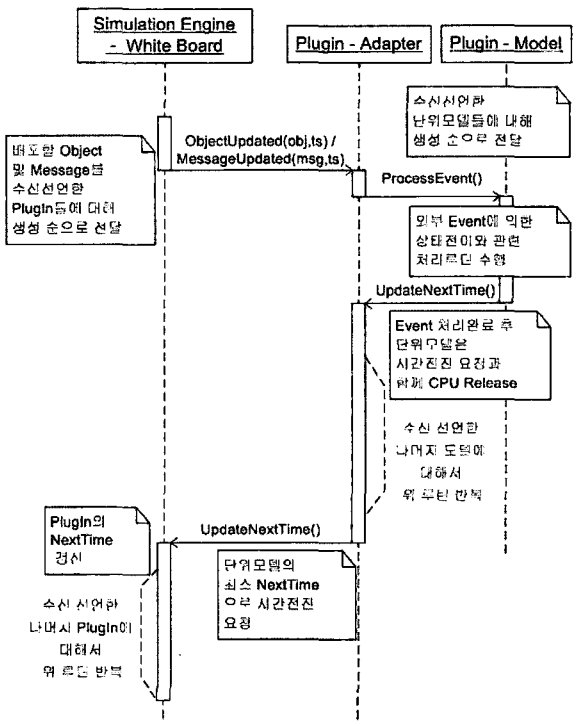
PlugIn Adapter에 의한 PlugIn 내부 단위모델에 대한 작업 배정이 수행되고 생성된 이벤트의 배포가 PlugIn 내부에 대해서 모두 이루어진 경우 PlugIn Adapter는 Inter-POM에 의해 PlugIn 외부로 배포 선언한 사건들에 대해서 시뮬레이션 엔진의 WB에 비동기식으로 일괄 갱신시키고 각 단위모델의 갱신된 NextTime 중 최소값으로 PlugIn의 NextTime를 정하여 시뮬레이션 엔진에 UpdateNextTime을 호출함으로써 자신의 작업을 중단한다.

시뮬레이션의 시작은 StartSim에 의해 이루어지며 이때의 시뮬레이션 시각과 각 PlugIn 및 내부 단위모

델들의 NextTime은 모두 0인 상태이다. 따라서 작업 배정은 PlugIn과 내부 단위모델의 생성 순위에 의해 최초 배정되며 이 후에는 발생되는 사건과 NextTime이 결정되는 결과에 따라 작업이 배정된다.

사건에 의한 방식은 시간전진에 의해 시뮬레이션이 진행되는 과정에서도 일부 설명되었지만 그림 6에 나타낸 바와 같이 시뮬레이션 엔진은 WB에 배포할 사건이 있는 경우 현재의 시뮬레이션 시각을 Time Stamp로 하여 FIFO(First-In-First-Out) 방식으로 해당 이벤트를 수신하는 각 PlugIn 들에게 저장된 사건들을 동기방식에 따라 순차적으로 전달(Forward)한다. ObjectUpdated와 MessageUpdated를 통하여 이벤트를 수신함으로써 작업배정을 받은 PlugIn은 수신한 Time Stamp로 자신의 시뮬레이션 시각을 전진시키고 PlugIn Adapter에 의해 해당 사건을 수신하는 단위모델들에게 ProcessEvent를 통하여 이벤트를 순차적으로 전달한다.

ProcessEvent에 의해 사건수신과 작업배정을 받은 단위모델은 시간전진과 상태갱신 등 관련처리루틴



[그림 6] 사건에 의한 스케줄링 및 동기화

을 수행하며 아울러 발생되는 출력사건들을 PlugIn Adapter의 WP에 비동기식으로 전달한다. 이러한 일련의 단위모델 상태갱신 중 시간전진이 필요하게 되는 경우 단위모델은 UpdateNextTime을 통하여 PlugIn Adapter에 시간전진을 요청함으로써 자신의 처리를 중단한다.

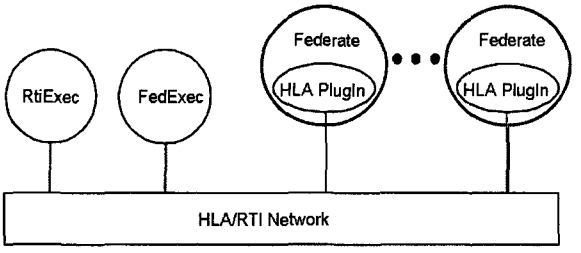
이와 같이 PlugIn Adapter는 외부에서 수신한 사건에 대한 배포와 자체 단위모델의 갱신 중 발생한 사건의 PlugIn 내부 배포가 완료된 경우 PlugIn 외부로 출력할 사건들을 UpdateObject와 UpdateMessage를 통하여 WB에 일괄적으로 전달한 후 단위모델들이 요청한 시간전진 시각들 중 최소값을 PlugIn의 NextTime으로 하여 전역 시뮬레이션 엔진에 시간전진을 요청함으로써 자신에게 배정된 작업을 중단한다.

PlugIn과 이들 PlugIn을 구성하는 단위모델에 대한 작업배정은 모두 시간전진과 사건전달에 의해 동일한 방식으로 이루어진다. 시뮬레이션 엔진의 WB와 PlugIn Adapter의 WP에 배포할 사건이 있는 경우 이를 수신하는 PlugIn 및 단위모델에 대한 배포에 의해 시간할당이 이루어지며 배포가 모두 이루어진 후에는 각 단위모델과 PlugIn의 시간전진 요청 중 최소값으로 시뮬레이션 시각의 전진을 허용함으로써 작업배정과 시뮬레이션의 진행을 동시에 이룰 수 있다.

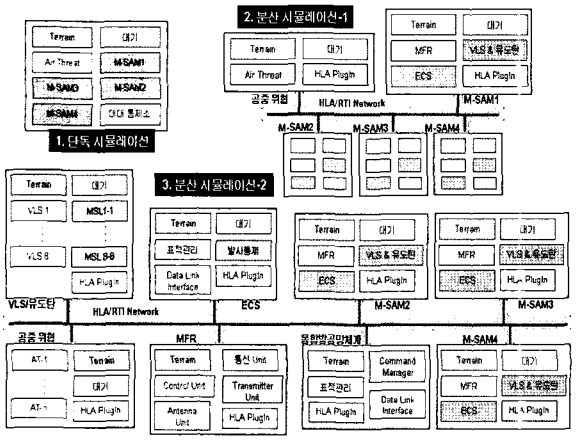
6. HLA 연동

PBA는 단일 노드에 의한 단독형(stand alone) 시뮬레이션을 뿐만 아니라 HLA를 통한 분산 연합 시뮬레이션을 위한 프레임워크를 제공한다. 따라서 그림 1과 그림 7에 나타낸 바와 같이 단독 시뮬레이션 시험과 함께 연동데이터 모델에 따라 HLA PlugIn만을 추가적으로 'Plug In' 함으로써 쉽게 연합 시뮬레이션을 구성할 수 있다.

그림 8은 중거리 지대공 유도무기에 대한 단독형 시뮬레이션과 분산 시뮬레이션 구현 시 적용 가능한 PlugIn 컴포넌트 구성을 예시로 나타내었다. 단독 시뮬레이션에서는 집약도 수준(Aggregation Level)을 높여 유도무기 체계 전체를 하나의 PlugIn에서 모의



[그림 7] HLA 분산 시뮬레이션



[그림 8] 중거리 지대공 유도무기체계 모의구성 예시

할 수 있도록 구성할 수 있으며 집약도 수준을 낮춰 각 하부 체계를 각각 PlugIn에 대응시킨 후 이를 시뮬레이션 노드로 할당하여 HLA 분산 시뮬레이션을 구성할 수도 있다.

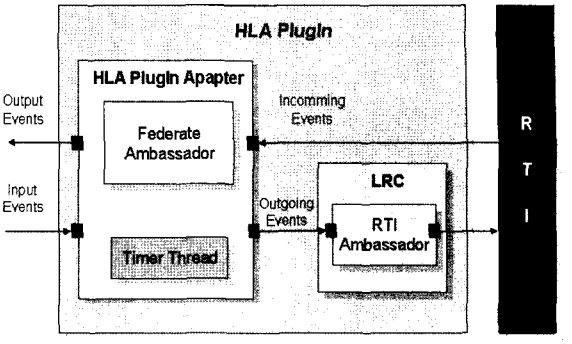
가. HLA PlugIn

HLA PlugIn은 단독 시뮬레이터가 HLA 분산 시뮬레이션 환경에 참여할 수 있도록 하는 PlugIn으로서 HLA 인터페이스를 위한 서비스 프레임워크이다.

그림 9에 나타낸 바와 같이 HLA PlugIn은 시뮬레이션 엔진의 작업배정과는 별도로 자체 Timer를 통해 연동주기에 따라 단독 시뮬레이터와 Run Time Infrastructure(RTI)간의 송수신 사건을 중계하며 분산 노드간의 동기화를 위한 시간관리 기능을 포함한다.

나. HLA 환경하의 자료교환

HLA PlugIn은 생성 시 RTI에 가입을 대행하고



[그림 9] HLA PlugIn 구조

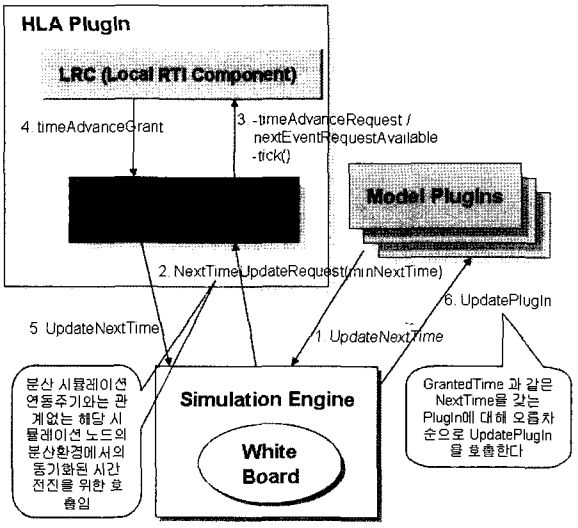
Federate Object Model(FOM)[8]에 따라 RTI로부터 수신하는 사건에 대해서 WB에 배포선언을 하고 WB로 부터 수신하는 사건에 대해서는 RTI로 배포선언을 함으로써 객체관리를 위한 초기화를 수행한다.

시뮬레이션 수행 시에는 WB보드로 부터 수신하는 사건에 대해서 바로 RTI Ambassador에 바로 전달하고 Timer에 의해 작업을 할당받아 축적된 출력 사건들을 RTI로 전달하며 RTI로 수신된 사건들에 대해서는 일괄적으로 즉각 WB에 이를 갱신함으로써 RTI와의 자료교환을 수행한다.

다. HLA 환경 하의 동기화

시뮬레이션 엔진은 PlugIn들과 PlugIn 을 구성하는 단위모델들에 대한 시간동기화를 수행하며 HLA를 통한 분산 연동 환경 하에서는 연합 시뮬레이션에 참여하는 각 Federate 간의 시간 동기화가 추가적으로 요구된다. 이를 위해 HLA PlugIn은 HLA 연동 창구가 되는 RTI의 시간관리 서비스를 통하여 동기화를 구현한다.

분산 시뮬레이션의 경우 시뮬레이션 엔진은 WB에 배포할 이벤트가 없는 경우 PlugIn의 NextTime 중 최소값으로 HLA PlugIn의 NextTime 갱신요청을 하게 되며 HLA PlugIn은 이를 받아 RTI에 해당시각으로 전진을 요청한다. 요청이 된 후 RTI에 의해 시간전진이 승인될 때 HLA PlugIn은 자신의 NextTime이 갱신되었다는 것 즉, 연합시뮬레이션의 갱신 시각을 시뮬레이션 엔진에 알려줌으로써 시간전진을 할 수 있도록 한다. 그림 10은 이러한 HLA 연동 시 동기화 과정을 나타낸다.



[그림 10] RTI 시간 동기화

7. 맺음말

본고는 M&S 구성 요소에 대한 재사용성을 높일 수 있는 공통 하부 기반에 대한 연구결과로서 PBA를 제안하였다.

PBA는 재사용 가능한 모델 구성품인 PlugIn이 시나리오 구성 시와 아울러 모의 실행 시 동적으로 생성('Plug-In')될 수 있는 하부 기반구조를 갖으며 관련 서비스를 제공한다.

또한 단독 시뮬레이션 시스템 개발을 위한 프레임워크를 제공할 뿐만 아니라 단독형으로 개발된 시스템에 대해서도 HLA를 통한 분산 시뮬레이션으로의 확장을 용이하게 할 수 있는 구조와 기반 서비스를 갖는다.

본 연구 결과를 바탕으로 아키텍처 프레임워크의 개발이 추후 과제이며 M&S 업무 적용 시 효율성 향상을 가져다 줄 것으로 판단된다.

참 고 문 헌

[1] Simulation Based Acquisition Industry Steering Group(SBA ISG), "SBA Functional Description - Version 1.1", 1999. 2.

[2] IEEE Std 1516-2000, IEEE Standard for Modeling and Simulation(M&S) High Level Architecture(HLA) - Framework and Rules.

[3] LCDR Doug Buchy, "Joint Modeling and Simulation System", Joint Modeling and Simulation System Program Office, 2000. 1.

[4] Joint Modeling and Simulation System Program Office, "JMASS Developer's Manual", 2003. 11.

[5] Robert J. Meyer, "Joint Modeling and Simulation System(JMASS)-What it does,... and What it doesn't!".

[6] Len Bass and Paul Clements and Rick Kazman, "Software Architecture in Practice", Addison-Wesley, pp.101~102, 1998.

[7] B. P. Zeigler, H. Praehofer and T. G. Kim, "Theory of Modeling and Simulation, 2nd ed.", Academic Press, 2000.

[8] IEEE Std 1516.1-2000, IEEE Standard for Modeling and Simulation(M&S) High Level Architecture(HLA) - Federate Interface Specification.