HLA 패더레이트 개발을 위한 ROM 프레임워크 설계 및 구현

김 대 석[†] · 배 종 환[††] · 류 재 철[†††]

요 약

패더레이션의 개선 가능성은 구성된 멤버 패더레이트들에게 유연성과 적응성을 요구하게 된다. 더욱이 비 연동화 모델을 HLA(High Level Architecture) 패더레이트로 개발하고 이를 가변적인 특정 패더레이션에 연동되도록 하기 위해서는 더 많은 시간과 노력이 요구된다. 본 연구에서는 이러한 문제를 해결하는 방법으로 ROM(RTI Object Model) 프레임워크를 설계하고 구현하는 방법을 제시하였다. ROM은 RTI(Run-Time Infrastructure) 프로그래밍과 패더레이트 시뮬레이션 프로그래밍을 완벽하게 분리시킴으로써 가변성 있는 FOM을 지원할 수 있는 HLA 패더레이트 개발을 비용과 생산성 측면에서 획기적인 효율을 제공하게 되었다. 즉 ROM은 RTI와 패더레이트 사이에 RTI 서비스를 관리하는 관리 계층과 실제로 객체 및 상호작용을 갱신 또는 반영하는 Foundation Class 계층을 두어 패더레이트 개발자들에게 보다 일반화된 HLA 서비스 사용환경을 제공해주고 동시에 반복적이고 하위수준의 RTI 프로그래밍을 자동화 할 수 있게 하였다.

A Design and Implementation of ROM Framework for Developing HLA Federate

Dae-seog Kim[†] · Jong-hwan Bae[††] · Jae-cheol Ryou[†††]

ABSTRACT

Possibility of federation improvement requires flexibility and adaptability of member federates. Moreover, to develop and convert a non-HLA (High Level Architecture) compliant model as a HLA federate and allow this federate to be integrated with a changeable federation, more time and efforts will be necessary. In this research, I proposed a method to design and implement a ROM (RTI Object Model) Framework as a solution to this problem. ROM completely separates RTI (Run-Time Infrastructure) programming and simulation programming therefore providing epochal efficiencies in cost and productivity to the development a HLA federate that supports a changeable FOM. That is, ROM contains two layers : 1) a management layer that manages RTI services between the RTI and the federate and 2) a Foundation Class layer that actually updates/reflects objects and interactions. These two layers allow federate developers to use more generalized HLA services and automates the iterative, low-level RTI programming process.

키워드 : 프레임워크(Framework), 패더레이트(Federate), HLA, ROM, RTI

1. 서 론

1989년 베를린 장벽의 붕괴와 함께 시작된 냉전시대의 종식으로 조성된 국제적인 화해 및 군축분위기는 대규모 야외기동훈련을 제한했을 뿐만 아니라 냉전시대에 책정되었던 비용을 줄이도록 요구하는 엄청난 압박으로 작용하게 되었다. 또한 냉전종식과 함께 국제적인 역학관계의 변화와 국지분쟁, 테러 등 새로운 도전과 불확실성은 군에 새로운 전략적 사고를 필요로 했다. 이러한 불확실성은 군에게 있어 보다 높은 적응력과 변화에 대해 유연하고 유능한 학습조직으로써의 능력을 갖추어야 한다는 전략적 목표를 설정하게 하였으며, 그 열쇠는 다름 아닌 끊임없는 훈련임을 인식하게 되었다[1].

그러나 많은 국가들이 급변하는 국제정세와 예측 불가능한 미래 전략환경에 부합되고, 보다 경제적이며 과학적인 군사훈련 수단으로 시뮬레이션을 선택하여 심혈을 기울여 발전시켜 왔으나, 다양한 훈련목적에 맞는 시뮬레이션 모델의 부족과 모델간 연계 운용성의 미흡 등으로 인해 투자대비 성과는 만족할 만한 수준이 되지 못하고 있었다.

1990년대 이전까지 개발된 군사용 시뮬레이션 모델들은 상호협력 및 연계 없이 수행되어 원래 목적한 이외 분야에 활용하기 위한 새로운 개념모델, 소프트웨어의 변경 등에 보다 많은 노력과 시간이 요구되었다. M&S(Modeling and Simulation)에 대한 이러한 문제점을 해결하기 위해 미 국방성은 시뮬레이션 상호운용성 보장을 위한 표준연동구조(HLA : High Level Architecture)를 제안하였고[2], 이는 2000년 9월에 IEEE1516 국제표준으로 등록되었다.

† 정 회 원 : 충남대학교 대학원 전산과학과
†† 정 회 원 : 제1정보통신 소프트웨어 개발부
††† 종신회원 : 충남대학교 전산과학과 교수
논문접수 : 2002년 8월 22일, 심사완료 : 2002년 11월 25일

본 연구에서는 이미 개발되어 운용중인 비표준연동 모델을 국제표준연동구조인 HLA 기반 시뮬레이션 모델로 전환 개발하는데 필요한 효과적이고 신뢰성 있는 방법으로 HLA 연동화 모델 공통구조(HLA Framework) 설계를 제안함으로써 이미 개발된 시뮬레이션 모델들을 보다 쉽게 HLA 기반 연동화 모델로 전환하여 새로운 목적의 분산 시뮬레이션을 구축할 수 있도록 하였다. 제안한 공통구조는 개발지원도구로 구현하였으며, 이를 창조21 연동화 모델 전환개발에 성공적으로 적용하여 그 유용성을 검증해 보았다[3].

본 논문은 2장에서 국제표준연동구조인 HLA 전반에 관한 개념을 설명하고, 3장에서는 제안한 프레임워크(Framework) 기반 HLA 패더레이트(Federate) 개발방법을 제시하며, 4장에서는 프레임워크 구현구조와 지원도구 구현에 대해 설명한다. 5장에서는 제안한 HLA 공통구조 프레임워크를 창조21 연동화 모델 개발에 적용한 사례연구 결과를 분석하고 마지막으로 6장에서 결론 및 향후과제를 제시한다.

2. 관련 연구

2.1 HLA(High Level Architecture) 개념

시뮬레이션 표준연동구조인 HLA(High Level Architecture)는 1990년대 중반 미 국방성에서 모든 유형의 시뮬레이션, 무기체계 및 C4I체계간의 상호운용성을 보장하고, 각 모형간의 재사용성을 향상기키기 위해 개발한 표준 분산 시뮬레이션 아키텍쳐이다.

분산 시뮬레이션은 최초 개별 시뮬레이터를 활용한 장비조작 숙달 훈련의 한계를 극복하기 위해 개발된 1세대 분산 시뮬레이션 구조인 SIMNET(SIMmulation NETwork) 체계로 부터 다양한 시뮬레이션 모형과 시뮬레이터를 하나의 동일한 가상전장환경으로 연동하는 2세대 분산 시뮬레이션 구조인 DIS (Distributed Simulation)로 발전하였고, 동시에 전구급 수준의 대규모 합동 시뮬레이션을 위한 분산 시뮬레이션 하부구조 및 통신규약을 정의한 ALSP(Aggregated Level Simulation Protocol)체계로 각각 발전해 왔다[4].

그러나 이러한 과거의 분산 시뮬레이션 구조는 많은 발전에도 불구하고 여전히 모형간 또는 체계간 연동소요는 단위모델의 작은 변경에도 불구하고 복잡한 연동화 작업이 발생하는 문제점을 내포하고 있었다.

DIS(IEEE1278)의 복잡도(C)는 다음과 같이 평가된다.
Let,
n : 연합체를 구성하는 단위모델 수
Fi : 연동 대상 모델중 i번째 모델
Li : Fi가 갖는 타 모델과의 연동 링크 수
$C = \sum(Fi \times Li)$, i = 1 to n 복잡도는 $O(n^2)$으로 나타난다.

그러나 HLA 분산 시뮬레이션 구조는 RTI(Run-Time Infrastructure)라고 하는 브로커 체계를 기반으로 함에 따라 연합체 구성 모델간의 연동구조 복잡도를 다음과 같이 현저히 감소시켰다.

HLA(IEEE1516)의 복잡도(C)는 다음과 같이 계산된다.
$C = \sum Fi$, i = 1 to n, 복잡도는 O(n)로 줄어든다.

HLA는 기본적으로 하나의 시뮬레이션 시스템으로는 모든 사용자들의 요구사항을 일거에 충족시키기 어렵다는 사실과 특히 사용자별 시뮬레이션 수준에 대한 관심과 충실도 요구가 상이하다는 것을 전제로 설계되었다.

또한 한 분야의 단일 시뮬레이션 개발자 집단으로는 모든 시뮬레이션 영역모의의 충실도를 보장하기 어려우며 미래 모든 시뮬레이션들의 유용한 결합 방법에 대해서도 예측하기 어렵다는 점 때문에 모델간의 상호운용성 및 컴포넌트들의 재사용성을 보장하고 개별 시뮬레이션들의 확장성 및 이식성을 확보하도록 설계하였다.

2.2 HLA 패더레이션(Federation) 구성

HLA 표준기술구조를 기반으로 개발된 개별 분산시뮬레이션 체계들은 상호연동하여 하나의 시뮬레이션 목적을 위해 하나의 패더레이션을 구성하게 된다. 이러한 시뮬레이션 연합체는 분산 시뮬레이션 체계인 패더레이트(Federate)와 연동기반체계인 RTI(Run-Time Infrastructure), 그리고 개별 시뮬레이션 체계들간의 상호교환해야 할 자료유형 및 관계성을 기술하고 있는 FOM(Federation Object Model)으로 구성된다[4-6].

2.2.1 패더레이트(Federates)

패더레이트는 개별 시뮬레이션 체계 또는 도구 등으로써 패더레이션 구축을 위한 연동체계인 RTI와의 단일 연결지점을 제공한다. 이러한 기능을 수행하는 각 패더레이트들은 라이브러리 객체나 루틴이 아닌 개별적으로 수행 가능한 완전한 실행 프로그램이다.

2.2.2 RTI(Run-Time Infrastructure)

시뮬레이션 연동체계인 RTI는 HLA 인터페이스 명세를 구현한 소프트웨어로서 패더레이션을 구성하고 있는 멤버 패더레이트간의 상호연동을 위한 서비스를 제공한다. RTI는 패더레이션별로 단지 하나만 존재하며 패더레이션 초기화 환경 파일인 RID(RTI Initialization Data)를 참조하여 수행되는 RTIExec(RTI Execution) 실행프로그램에 의해 가동된다. RTI는 모든 HLA 서비스 루틴을 포함하고 있어 개별 패더레이트들은 반드시 RTI를 통해서만 연동서비스를 제공받을 수 있도록 하고 있다.

2.2.3 패더레이션 객체모델(FOM : Federation Object Model)

RTI가 개별 패더레이트들에게 HLA 연동서비스를 제공하는 역할을 수행한다면 FOM은 개별 시뮬레이션 패더레이트들이 상호 연동하고자 하는 데이터를 정의하고, 개별 연동데이타별로 수행되어야 할 서비스들을 기술한 표준화된 문서라고 할 수 있다. 이러한 FOM은 표준화된 규약에 의해 생성되

고 이는 FED(패더레이션 Execution Data) 파일로 생성되어 궁극적으로 FedExec(Federation Execution)에 의해 수행된다.

2.2.4 시뮬레이션 객체모델(SOM : Simulation Object Model)

각 시뮬레이션 패더레이트의 내부 오퍼레이션에 초점을 맞추어 작성된 시뮬레이션 능력을 표현하는 것으로서 패더레이션 구성을 위해 타 시뮬레이션 패더레이트가 사용 가능한 객체 및 상호작용을 표현하고 있다. SOM은 패더레이트당 하나만 존재하며, SOM에 표현되지 않은 자료에 대해서는 타 패더레이트에서 사용할 수가 없다.

2.3 HLA 구성요소

표준연동체계인 HLA는 분산 시뮬레이션의 개념적 표준 기술구조로 HLA 규칙(Rules), OMT(Object Model Template), Interface Specification으로 구성된다. 이는 모든 HLA 기반 시뮬레이션 체계들이 반드시 준수해야 할 표준을 위한 하나의 틀을 의미한다.

2.3.1 HLA 규칙(Rules)

HLA 규칙은 패더레이션 실행시 각 시뮬레이션들 사이에 일어나는 상호작용들에 대해 각 구성요소가 수행해야 할 사항들을 정의한 것으로 패더레이션 및 패더레이트 규칙이 각각 5개로 구성되어 있다[7].

2.3.2 객체모델 형판(OMT : Object Model Template)

객체모델 형판(OMT)은 패더레이션과 패더레이트가 자신의 클래스와 속성, 상호작용 등을 표현하기 위해 사용하는 공통의 표준화된 자료형식을 규정하고 있는 프레임워크이다.

OMT는 표준화된 자료형식을 적용함에 따라 다양한 플랫폼에서 개발된 분산 시뮬레이션들의 상호교환 자료들을 별도의 인터페이스 작업없이 필요한 패더레이트에게 전달할 수 있고, 패더레이션 실행종료 이후에는 자료의 재사용이 가능하도록 되어있다.

이러한 OMT는 객체 클래스 구조 및 상호작성, 속성, 파라메타, 자료사전 등을 정의할 수 있다. 패더레이션 수행에 필요한 FOM이나 패더레이트 표현능력을 나타내는 SOM 등은 모두 OMT에서 정의한 자료표현 형식으로 만들어진다[8].

2.3.3 인터페이스명세(Interface Specification)

인터페이스 명세는 패더레이트가 패더레이션과 어떻게 상호작용을 하고 궁극적으로 패더레이션내 다른 패더레이트들과 어떻게 상호작용을 수행하는지를 정의해 준다. 인터페이스 명세는 RTI 라이브러리로 구현되어 멤버 패더레이트들에게 상호연동에 필요한 서비스를 제공함으로써 전체 패더레이션의 분산운영을 지원한다. 각 패더레이트들은 RTI가 제공하는 서비스를 통해서 자신의 변경된 연동 클래스 속성이나 상호작용들을 다른 패더레이트에게 전달한다.

이러한 패더레이트간의 자료분산은 반드시 RTI를 통해서만 수행되며, RTI는 인터레이스 명세 수행을 위한 6가지 서비스로 패더레이션 실행을 제어하는 패더레이션 관리, 자료교환의 협상을 위한 선언 관리, 객체 및 상호작용의 생성, 갱신, 삭제 등을 위한 객체 관리, 속성 소유권 이양/획득을 위한 소유권 관리, 패더레이션 시간 동기화를 위한 시간 관리, 자료갱신, 수신 영역 설정, 변경, 삭제를 위한 자료 분배 관리영역으로 구분되어 진다[9].

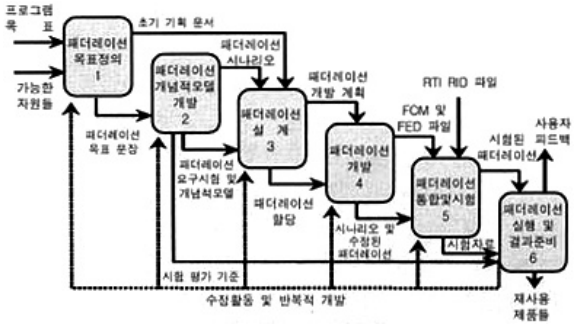
2.4 패더레이션 및 패더레이트 개발 방법

HLA는 근본적으로 분산 시뮬레이션을 구현하기 위한 개념적 표현으로 궁극적으로는 여러개의 개별 패더레이트들을 원활하게 통합시켜 하나의 새로운 목적을 위해 수행되는 패더레이션을 구축하는 표준연동기술 기반구조이다. 따라서 앞 절에서 언급한 HLA 규칙이나 객체모델 형판 또는 인터페이스 사양 등은 보다 용이하게 패더레이션을 구축할 수 있도록 하기 위함이다.

2.4.1 패더레이션 개발 방법(FEDEP)

패더레이션 개발은 IEEE12207로 표준화된 소프트웨어 개발 방법론만을 단순히 적용해서는 안된다. 왜냐하면 패더레이션은 단지 프로그램내의 기능적 프로세스 분산 형태가 아닌 물리적으로 완전히 독립된 패더레이트 소프트웨어를 표준연동기술 기반구조로 통합시키는 모양으로 구성되기 때문이다.

이러한 패더레이션 개발의 특성을 고려해서 제안된 것이 (그림 1)과 같은 패더레이션 개발 절차(FEDEP : Federation Development and Execution Process)이다[10].



(그림 1) FEDEP 모델

이러한 패더레이션 개발 절차에도 불구하고 패더레이션 개발은 설정된 패더레이션 목표별 재사용 가능한 패더레이트의 가용유무에 따라 세부적으로는 아주 다른 내부 프로세스 과정을 거치게 된다. 다음은 패더레이트 가용여부에 따라 분류한 패더레이션 개발방법이다.

첫째 : 멤버 패더레이트 및 패더레이션을 완전히 새로 개발하는 방법

둘째 : 개발된 패더레이션에 새로운 멤버 패더레이트를 추가하는 방법

셋째 : 기개발된 모델들을 재사용하여 새로운 패더레이션을 개발하는 방법

위에서 제시된 패더레이션 개발방법은 결과적으로 패더레이션 객체모델 개발과정으로 구체화된다. FOM 개발은 FEDEP에서 정의된 바와 같이 목표를 기초로 실시한 개념적 분석으로 패더레이션을 위한 필요 모델을 선정하고 각 모델별 기능을 할당함으로써 시작된다. FOM은 표준화된 객체모델형판을 이용하여 모델별로 할당된 기능들간에 필요한 상호작용을 설계하고 정의하는 과정으로 다음과 같은 접근방법들이 있다.

① 페더레이션 시나리오, 개념모델 등으로부터 "Bottom up" 방법의 FOM 구축
② 멤버 페더레이트 SOM을 병합하고 관심영역에 포함되지 않는 SOM의 특징 제거
③ 원하는 FOM과 가장 근접한 SOM으로 시작하여 관심영역에 적용되지 않는 SOM 특징들을 제거하고 추가로 요구되는 다른 SOM의 요소들을 병합
④ 유사한 응용의 기존 FOM으로 시작하여 필요한 사항을 수정하고 확장
⑤ 공통 참조용 FOM(RFOM)으로 시작하여 응용에 불필요한 FOM 요소들을 제거하고 필요시 수정 및 확장

2.4.2 패더레이트 개발 방법

패더레이트 개발은 기존의 시뮬레이션 모델개발과는 다소 상이한 의미를 갖는다. 즉, 패데레이트란 기존 시뮬레이션 모델에 표준연동구조가 적용된 HLA 연동화 모델을 의미한다. 이러한 패더레이트 개발은 이미 개발되어 사용중인 기존의 시뮬레이션 모델을 전환하여 개발하는 방법과 패더레이션 개념모델에서 할당된 기능을 수행하는 새로운 연동화 모델로 처음부터 개발하는 경우로 구분될 수 있다.

두가지 방법은 공통적으로 기존의 시뮬레이션 개발자들이 새로운 표준연동구조에 관한 기술적 이해를 필요로 한다. 더구나 이러한 표준연동 기술구조는 단지 프로그램을 구현하는 차원이 아닌 표준화의 의미로 인해 개발자들이 약 130여종에 이르는 세부 표준서비스에 대한 기술적 이해와 구현능력을 구비해야 한다.

OMT와 RTI는 이러한 표준서비스를 구현하는데 필요한 많은 라이브러리와 표준화 요소를 제공하지만 패더레이트들이 HLA 규칙과 서비스들을 위해 필요한 관리수준의 작업들은 여전히 패더레이트 개발자들에게 요구하는 기술적 어려움들로 존재한다.

3. ROM(RTI Object Model) 프레임워크 설계

프레임워크 기반 HLA 패더레이트 개발방법은 1998년 Kevin Cox에 의해 제안된 바 있다[6-8]. Cox는 HLA 패더레이션을 개발하는 과정에서 인터페이스 사양들을 구현하는데 필요한 개발자들의 노력과 패더레이트별로 소요되는 작업의 중복성을 경험하고 이를 해결하기 위한 접근으로 프레임워크 개발을 시도했다.

그러나 본 연구에서는 패더레이션을 구축하는 것이 아니라 이미 구축되었지만 완성도가 떨어지거나 변경가능성이 높은 패더레이션에 새로이 멤버 패더레이트를 추가하는데 필요한 프레임워크 구조를 제안하였다.

일반적으로 패더레이션을 개발하는 경우에는 FEDEP의 개념모델 분석단계에서 이미 어느정도 패더레이션 객체모델이 확정적이며 각 패더레이트는 확정된 패더레이션 객체모델을 기초로 연동화 모델을 개발하게 된다. 하지만 이미 구축된 패더레이션에 패더레이션 개발팀과는 구분된 별개의 팀에 의해 새로운 멤버 패더레이트를 추가하는 경우에는 다음과 같은 사항을 고려해야 한다.

첫째 : 패더레이트 추가에 따른 기존 패더레이션의 변경은 최소한으로 요구되어야 한다.
둘째 : 추가될 패더레이트는 이미 개발된 모델이거나 새로 개발해야 할 모델일 수 있다.
셋째 : 새로운 모델을 개발하는 경우, 시뮬레이션 개발자들에게 HLA 인터페이스 사양 구현에 별도의 노력이 요구되지 않아야 한다.

일반적으로 HLA는 데이터와 구조를 분리하고 있으며, 이것은 OMT 표준에 따라 정의된 OMT 객체와 상호작용이 HLA 기반 소프트웨어의 수정 없이 구축되고 교환될 수 있음을 의미한다. 하지만 LibRTI로 실체화된 인터페이스 명세를 제외한 HLA 규칙 및 OMT의 구현 및 적용은 여전히 프로젝트를 수행하는 개발자의 몫이다.

OMT에 의해 추상화된 객체는 각 패더레이트에서 다양한 방법으로 실체화되고 이들은 HLA 규칙을 준수하기 위한 공통 기능성 혹은 서비스를 필요로 하게 된다.

SOM에서 정의된 모든 객체 및 객체 속성들은 FOM에서 정의된 추상객체 및 추상객체 속성과 매핑되고, 생성된 객체를 인터페이스 명세에 따라 RTI에 등록하고 갱신된 속성값을 전송할 것이다. 또한 타 패더레이트에 의해서 생성된 객체를 발견하여 객체를 생성하고, 생성된 객체의 속성 값을 RTI로부터 갱신 전달을 기대하게 된다.

예를 들어 창조21 모델로부터 지상 전투부대(지상 근접전투 클래스)의 인스턴스를 생성하는 경우를 보면, 창조21 모델은 FOM에 정의된 추상클래스(지상 근접전투 추상클래스) 의 핸들값을 가지고 RTI 서비스를 이용하여 인스턴스를 등록하고 이때 제공된 인스턴스 핸들값의 상태를 관리하게 된다. 또한 지상 전투부대의 속성 값을 갱신하고 이를 타 패더레이트에게 전송하기 위해서는 FOM에 정의된 추상 클래스 속성 값과 매핑을 하여 RTI를 통해 갱신정보를 전송하게 될 것이다.

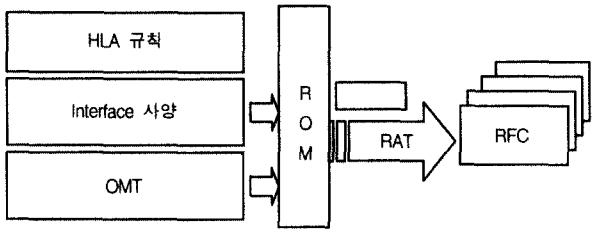
이러한 객체의 등록과 갱신은 부대객체 뿐 아니라 정의된 모든 객체에 적용되는 공통적인 특징이며, 이외 대부분의 HLA 서비스 수행이 유사하게 이루어진다. 이러한 공통기능 및 서비스를 통합 관리하고 일관성 있는 연동 구조를 제공하기 위해 프레임워크를 구상하게 되었다.

본 연구에서는 확장된 RTI로 표현할 수 있는 ROM 프레임

워크를 제안하고 있다. 기본적으로 RTI는 HLA 인터페이스 명세를 구현하고 있기 때문에 연동 인터페이스를 위한 서비스 이외의 패더레이트를 위한 객체의 상태 관리 및 데이터의 관리 방법 등 구현에 필요한 일련의 기능은 제공하지 않고 있다.

따라서 패더레이트 개발자들은 인터페이스 서비스 이외의 추가적인 데이터 관리를 위한 구현을 담당해야 하며 이는 모든 RTI 서비스에 대한 구체적인 이해를 요구하게 된다. ROM은 이러한 패더레이트 개발자들의 추가적인 구현 작업을 개선하는 방안으로 제안되었다.

ROM에서는 모든 RTI 서비스를 제공하여 RTI와 호환성을 유지하고 FOM 정보를 관리함으로써 데이터의 관리 및 변환을 자동화할 수 있도록 지원하며, HLA 규칙을 준수하기 위한 기능을 제공한다(그림 2).



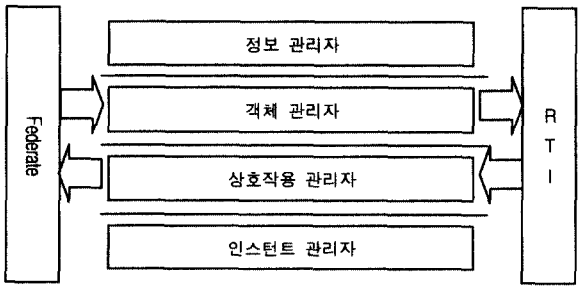
(그림 2) HLA 구성요소와 RTI 객체모델

3.1 ROM 프레임워크의 정의

ROM은 패더레이트와 RTI의 인터페이스를 관리하고 RTI의 공통서비스 기능을 제공하기 위한 Function 집합으로 RTI에서 제공하지 않는 각 패더레이트들의 오브젝트에 대한 생성, 갱신, 삭제 정보를 관리하고, 패더레이트의 연동 오브젝트 및 상호작용 정보를 타 모델로 전송하며 패더레이트 차원에서 요구되는 타 패더레이트의 연동 오브젝트 및 상호작용을 생성하고 관리한다.

3.2 ROM의 구성요소 및 기능

ROM의 구성요소는 (그림 3)와 같이 정보관리, 객체관리, 상호작용관리, 인스턴스관리의 4개 관리 기능으로 구성되어진다.



(그림 3) ROM 구성

3.2.1 정보관리자(Information Manager)

정보관리자는 패더레이션, 데이터 선언, 시간관리를 목적으로 하며 FOM에 정의된 객체 및 속성 정보를 관리한다. 이는 모든 패더레이트의 기본 기능인 패더레이션 관리, 선언관리, 시간관리를 모델과 분리하여 관리함을 의미한다. 또한 FOM에 정의된 객체 및 속성 정보를 관리함으로써 패더레이트 개발자에게 데이터의 변환 및 매핑 정보를 제공한다.

일반적으로 완전한 개념모델의 설계는 기대할 수 없지만 설령 개념 모델이 완전하다 해도 패더레이션에 참여하는 패더레이트의 상태에 따라 속성값의 갱신 및 반영에 대한 정보는 충분히 가변적이다. 따라서 패더레이션 개발자에게 모든 객체의 속성에 대한 갱신과 반영에 대한 구현을 구성되는 패더레이트의 상태에 따라 변경하거나 삭제하도록 강요하게 된다.

이는 잘못된 개념모델 설계 또는 성장하는 모델에 대한 반응이 상대적으로 늦어지거나 복잡해지는 원인을 제공하여 수정개선 및 유지보수를 어렵게 할 수 있다. 따라서 패더레이트 개발과 연동 환경(패더레이션에 가입할 것이지, 어떤 속성에 대해서 갱신하고 반영할 것인지, 시간정책은 어떻게 수립할 것인지)과는 분리되어야 하며 이러한 요구를 충족하기 위해 연동 정보관리자를 설계하였다.

3.2.2 객체관리자(Object Manager)

객체관리자는 객체의 상태 정보를 관리하기 위한 것으로 객체의 등록(Register)/발견(Discover) 관리, 객체의 소멸(Delete)/삭제(Remove)관리, 객체속성의 갱신(Update)/반영(Reflect) 관리를 목적으로 한다.

객체 관리자를 통하여 객체를 관리함으로 패더레이트의 객체와 연동 추상 객체와의 매핑과 열거형 및 복합형 데이터 변환 (인코딩 및 디코딩)으로부터 패더레이트 개발자를 자유롭게 하고 개발자가 시뮬레이션 고유의 기능에만 전념할 수 있도록 유도한다. 이는 정보관리자가 제공한 모든 종류의 일반 정보와 다양한 확장 정보를 객체 관리자가 활용할 수 있도록 지원한다.

만일 객체관리자와 같은 관리 기능이 없는 경우 패더레이트 개발자들은 객체를 생성하고 등록하기 위해서, 혹은 속성값을 갱신하고 전송하기 위해서 RTI의 모든 서비스를 사용할 수 있어야 하며, 데이터의 변환(인코딩, 디코딩)을 위해서 FOM에 정의된 모든 데이터 정보에 대해 관심을 가져야 한다. 여기에는 시간과 비용의 손실뿐만 아니라 상당한 소스 코드의 중복을 초래하게 되고 개발자의 취향이나 성향에 따라 데이터의 일관성을 상실하게 된다. 결국 개발이 지연되고 유지보수 및 변경개선을 어렵게 하여 패더레이션의 확장 및 유연성을 크게 떨어뜨리게 될 것이다.

객체관리자는 이러한 복잡한 문제를 단순화시킬 수 있다. RTI 서비스 및 데이터 변환 창구를 단일화하기 때문에 데이터의 일관성을 유지해줄 수 있고 데이터 변환기법의 변경 및 추가를 유연하게 대처하므로 개념모델 변경에 의한 패더레이션 변화에 쉽게 적응할 수 있는 기회를 제공한다.

3.2.3 상호작용 관리자(Interaction Manager)

상호작용 관리자는 상호작용 이벤트를 관리하기 위한 것

으로 패더레이트의 상호작용 이벤트를 RTI에 전송하고, RTI의 상호작용 이벤트를 수신하여 실행해 준다. 객체 관리자와 마찬가지로 나열형 및 복합형 매개변수 변환(인코딩 및 디코딩)으로부터 패더레이트 개발자를 자유롭게 하고 개발자가 시뮬레이션 고유의 기능에만 전념할 수 있도록 유도한다.

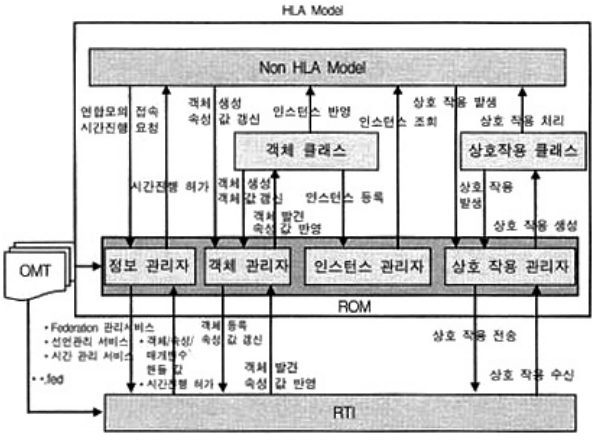
3.2.4 인스턴스 관리자(Instance Manager)

인스턴스 관리자는 객체 관리자에 의해서 생성된 객체의 인스턴스를 관리한다. 패더레이트 및 RTI로부터 등록된 객체는 객체 관리자에 의해서 추상 클래스의 인스턴스로 실체화되고 인스턴스 관리자에게 등록된다. 인스턴스 관리자는 생성된 객체에 접근할 수 있는 방법을 제시하며 RTI 및 패더레이트로 부터의 공통 요구(Refresh, SendAllData, UpdateRequest 등)에 응답한다.

3.3 RFC 구현구조 및 RAT 지원 도구

3.3.1 RFC(ROM Foundation Class) 구현 구조

RFC는 (그림 4)와 같이 ROM의 라이브러리를 활용하여 생성된 추상클래스 (연동 클래스)를 의미하는 것으로 RFC는 연동모델이 클래스를 ROM에 등록 요청함으로써 자동 생성되고 관리된다.



(그림 4) ROM Foundation Class Structure

RFC의 생성은 ROM의 모든 기능을 활용할 수 있음을 의미하며 패더레이트 개발자는 객체의 등록 및 갱신 정보의 등록 행위만으로 RTI에 데이터를 전송하기 위한 데이터의 적합성(Publish는 되어 있는지, 속성 핸들값이 무엇인지, 데이터를 어떻게 전송할 것인지, 어느 시기에 전송을 해야 되는지 등)에 대해 전혀 고려하지 않아도 된다.

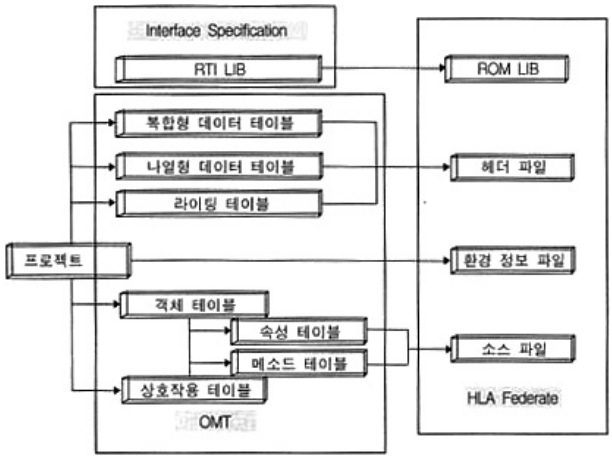
또한 RTI로부터 객체의 생성이 발견되면 자동으로 RFC를 생성하고 데이터의 갱신 정보를 반영해준다. 이는 RFC의 추상 객체가 FOM의 정보를 기반으로 생성되고 ROM의 연동 정보관리자에 의해서 관리되기 때문에 가능하다.

RFC는 이미 개발이 완료된 모델을 HLA 연동화 모델로 전환하기 위해 유용하게 사용 될 뿐만 아니라 새로운 HLA 패더레이트 개발에도 적합하게 사용될 수 있다.

RFC는 ROM으로부터 상속받는 구조가 아니라 ROM의 Lib를 활용하도록 설계하였다. RFC와 ROM은 FOM을 통하여 밀접하게 연관되어 있으며 이는 표준포맷 파일로 ROM에 등록된다.

3.3.2 RAT(RFC Automation Tool) 지원 도구

RAT은 개념모델에 의해서 도출된 추상적 객체를 실체화시키기 위한 개발지원 도구로 소프트웨어 개발자에게 객체지향적인 공통의 기본 코드를 제공하여 코드의 일관성을 유지하도록 지원하고, ROM과의 공통 인터페이스를 통한 HLA 연동성을 보장하고 지원하기 위한 개발 지원 도구이다(그림 5).



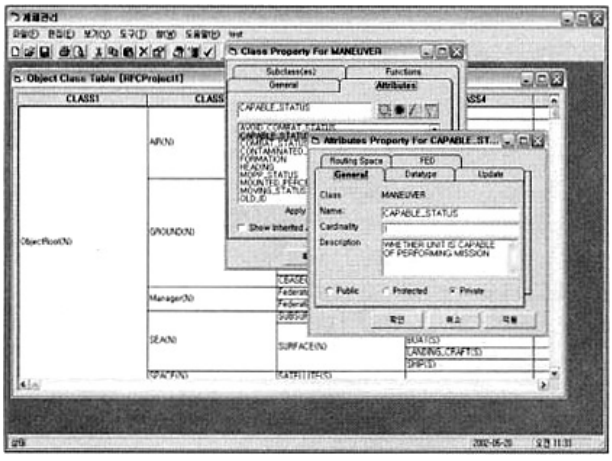
(그림 5) RAT 구조

RAT을 이용한 패더레이트 개발은 RAT을 이용하여 연동 객체 및 상호작용 클래스를 생성하고 이를 기초로 연동용 FOM이나 SOM을 생성할 수 있다. 만약 참조할 수 있는 적절한 FOM 또는 SOM이 존재한다면 추가될 연동 객체 및 상호작용 클래스만 다시 생성하여 새로운 목적의 FOM 또는 SOM을 생성할 수 있으며, 이러한 공통객체 및 상호작용 클래스를 이용하여 새로운 멤버 패더레이트를 개발하면 된다.

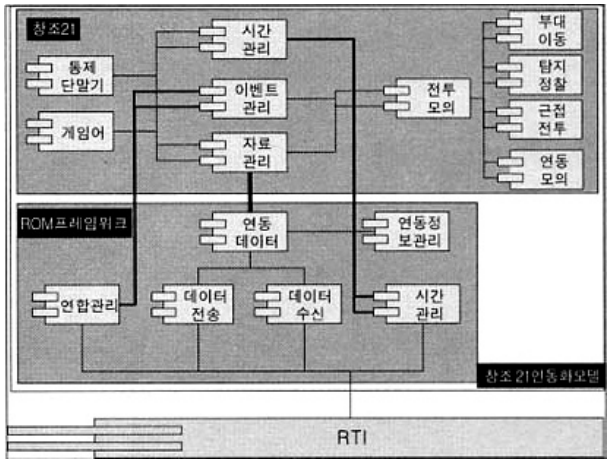
이와 같이 RAT은 시뮬레이션 개발자들에게 RTI의 인터페이스 서비스들의 이해 없이도 패더레이트를 개발할 수 있게 해주는 유용한 수단으로 다음과 같은 기능을 포함하고 있다.

① 외부 연동 데이터에 의한 연동 클래스를 생성한다.
② 연동에 필요한 각종 데이터(SOM, FOM, MOM)를 제공한다.
③ ROM의 lib를 활용한 인터페이스를 제공(연동 데이터의 전송 및 수신)한다.
④ 객체 클래스의 생성, 관리 및 편집기능을 제공한다.
⑤ 상호작용 클래스의 생성, 관리 및 편집기능을 제공한다.
⑥ 원시코드의 관리기능을 제공한다.

RAT은 OMDT의 모든 기능 이외에 객체 또는 상호작용 클래스의 소스 코드를 편집할 수 있는 기능을 제공한다(그림 6).



(그림 6) ART 메인화면 구성



(그림 7) 창조21 연동화 모델 구조

4. ROM 프레임워크 적용 사례

ROM 프레임워크를 사용하여 창조21 연동화 모델을 설계하고 이를 RAT을 이용하여 성공적으로 구현하였다.

창조21 연동화 모델은 이미 개발되어 사용중인 육군의 훈련 시뮬레이션 모델인 창조21 모델을 국제표준연동구조인 HLA 서비스를 수행할 수 있도록 개선한 HLA 패더레이트이다.

창조21 연동화모델은 기존의 창조21 모델의 시뮬레이션 측면의 기능적 요소와 HLA 패더레이트로서의 연동서비스 수행기능을 함께 포함하고 있다. 따라서 패더레이트로서의 창조21 연동화 모델은 하나의 SOM을 가지며, SOM에 표현된 객체 및 속성들의 갱신 또는 반영, 소유권 이전 서비스를 제공하며, 갱신을 위한 임계 조건을 변경할 수 있는 연동을 위한 특성을 내포하고 있다.

4.1 개발 환경

창조21 연동화 모델은 RTI 1.3NGv4 및 OMDT v1.3을 사용하여 개발하였으며, Redhat Linux 6.2 운영체제의 C++ 언어로 구현되었다. 특히 창조21 연동화모델은 자체 개발한 RFC(ROM Foundation Class) 자동화 도구인 RAT을 이용하여 연동화 모델의 연동서비스 코드 반복성을 제거하고 개발 효율성을 개선하였다.

4.2 창조21 연동화 모델 구현

창조21 연동화 모델은 기존의 창조21 모델의 수행기능을 보장하면서 연동에 필요한 연동객체 및 상호작용을 식별하여 SOM으로 등록했다. SOM의 연동객체들은 창조21 연동화 모델이 참가하게 될 기존 패더레이션 FOM에 포함되어 새로운 패더레이션을 구성할 수 있게 만들었다.

창조21 연동화 모델은 RTI를 통해 연동하게 될 각종 연동데이타 또는 상호작용의 처리를 직접 전달하거나 수신하지 않고 (그림 7)에서와 같이 ROM 프레임워크를 통해 수행한다.

예를 들어 창조21 연동화 모델에서 연동데이타의 처리절차는 다음과 같이 수행된다. 게임어나 통제단말기에 입력된 명령은 처리될 시간을 가지고 이벤트 관리 목록에 등록되고 이는 전투모의 엔진에 의해 처리된다. 이때 처리자료들은 자료관리 목록에서 관리되며 만약 이들이 연동데이타인 경우 ROM 프레임워크의 연동관리자에게 통보되어 연동관리자로 하여금 연동데이타 및 연동정보관리를 수행하고 아울러 RTI에게 해당자료의 갱신이나 전달을 통보한다.

연동데이타 갱신 또는 전달과 마찬가지로 타 패더레이트로부터 수신하는 연동데이터에 대한 처리방법도 유사하다. FOM에 의해 식별된 수신 연동데이타를 RTI로부터 핸들에 의해 받게 되면 ROM 프레임워크는 이를 연동정보 내에서 해당 오브젝트 또는 상호작용의 속성들을 각각의 관리자가 확인하여 해당 연동클래스에 반영하고 창조모델에 알린다.

이러한 ROM 프레임워크를 통한 연동절차를 수행함으로써 창조모델은 연동에 따른 변경 개선소요를 최소화시킬 뿐만 아니라 패더레이션 변경에 있어서도 프레임워크 레벨에서 대부분 수용 가능하기 때문에 유연성을 제공할 수 있게 되었다.

4.3 ROM기반 HLA 패더레이트 개발 효과

일반적인 방법으로 구현되는 HLA 패더레이트와 ROM 프레임워크를 이용한 개발 방법간의 차이효과는 Jacobson의 Software Reuse에서 밝힌 소프트웨어 프레임워크의 이점을 참고하여 간접적으로 확인할 수 있다.

Jacobson에 의하면 프레임워크는 제품출하까지 걸리는 시간을 2배에서 5배까지 줄일 수 있으며, 소프트웨어 결함 및 유지보수 비용을 5배에서 10배까지 줄일 수 있다고 했다.

결국 프레임워크 적용시 전체 소프트웨어 개발비용을 최소 25%에서 85%까지 절감할 수 있다고 한다[9]. 실제로 창조21 연동화 모델 개발에 있어서는 약 70%의 효과가 있었으며 이는 개발지원도구의 완성도가 높아지면 더욱 효과가 증가할 것으로 예측되었다.

5. 결 론

모든 기존 시뮬레이션 모델은 잠재적인 패더레이션 멤버라고 여겨져야 한다. 이러한 잠재적 멤버들은 실제로 패더레이션 멤버로 결정되는 순간 HLA 패더레이트로 전환해야 하는 문제를 똑같이 직면하게 되고 개발자들이 HLA의 기술적 이해를 수행하는데 상당한 시간과 노력을 요구 받게 될 것이다. 그러나 제안한 바와 같이 ROM 프레임워크를 사용할 수 있다면 모델을 패더레이트로 전환 개발해야 하는 개발자들은 더 이상 HLA의 기술적 난이도에 얽매이지 않아도 될 것이다[3, 11].

프레임워크를 이용한 HLA 패더레이트 개발자들은 먼저 잠재적 모델이 멤버로 참가하게 될 패더레이션의 개념적 모델을 재구성하고 해당 패더레이트가 수행해야 할 역할을 정의하는 수준에서 요구되는 SOM을 결정하고 개발해야 한다. 그리고 나서 프레임워크 개발도구를 활용하여 해당 SOM에 해당하는 연동 클래스들을 생성하고 이를 자신의 모델에 적용하여 재 컴파일하는 것만으로 새로운 목적하는 HLA 멤버 패더레이트를 개발할 수 있다.

여기서 각 모델이 연동데이타를 수신하여 표현해야 할 내부적 표현에 대해서는 언급하지 않았다. 이는 모든 패더레이트들이 연동 데이터의 모델링적 요소로 논문에서 제안하는 프레임워크를 이용한 HLA 패더레이트 개발의 범주를 벗어나는 것이다. 즉, 프레임워크를 이용한 패더레이트 개발이란 패더레이션 멤버로서 HLA 규칙과 OMT, 인터페이스 사양을 충족시키는 시뮬레이션 모델을 만드는 것 자체를 의미한다.

결론적으로 본 연구에서는 연동에 필요한 모든 데이터 관리의 일관성을 유지하여 데이터의 변화로부터 모델의 수정 소요를 최소화할 수 있는 메카니즘을 제공하였고, 공통 라이브러리(ROM)를 제공하여 각 패더레이트에서 구현 해야 될 기능들을 패더레이트로부터 분리해 내고, 패더레이션의 개념 모델 변경에 능동적으로 대처할 수 있도록 하였다.

연구결과로 제시된 ROM 및 RAT는 아직까지 일부 서비스들을 대상으로 구현한 것으로 앞으로 추가적인 HLA 서비스들을 계속 확장 구현해야 한다. 그러나 제안한 ROM 프레임워크는 비 HLA 연동화 모델을 보다 쉽게 표준화된 연동화 모델로 전환할 수 있는 유용한 기술적 구조를 지원하고 있다고 하겠다.

참 고 문 헌

[1] 고든 R.설리반, 마이클 V.하퍼, "장군의 경영학", 창작시대사, p.30, 1998.

[2] DoD, USD(A&T), "DoD Modeling and Simulation Master Plan," Washington, October, 1995.

[3] Dae-Seog Kim, Hae-Kwan Lee, Yong-Hyo Kim, "A Framework Approach to HLA Compliant Federate Development(CJ21_HCM : A Case Study)," The 4th ROK-US DM&S Workshop, May, 2002.

[4] 장상철, 손미애, 서혜숙, 위정현, "국방 모의분석체계 구축을 위한 상위체계 구조 기술연구", 한국국방 연구원, pp.25-27, 1999.

[5] 서혜숙, 김태윤, "HLA 기반 모형의 네트워크 트래픽 감소를 위한 모형 설계에 관한 연구", 정보처리학회 춘계학술발표논문집, 제9권 제1호, pp.1-2, 2002.

[6] Defense Modeling and Simulation Office, "HLA Federation Development and Execution Process(FEDEP) Model, Version 1.2 Draft," May, 1998.

[7] DMSO U.S. Dept. of Defense, "High Level Architecture Rules, Version 1.3," April, 1998.

[8] DMSO U.S. Dept. of Defense, "High Level Architecture Object Model Template Specification, Version 1.3," February, 1998.

[9] DMSO U. S. Dept. of Defense, "HLA Interface Specification, Version1.3," April, 1998.

[10] DMSO U. S. Dept. of Defense, "HLA Federation Development and Execution Process (FEDEP) Model, Version 1.5," December, 1999.

[11] Dae-Seog Kim, Hae-Kwan Lee, Jae-Cheol Ryou, "ROM Framework Approach to develop a HLA Federate for Multi-national Federation," Simulation Interoper-ability Workshop, September, 2002.

김 대 석

e-mail : daeseog@prumail.co.kr

1986년 육군사관학교 전산학 학사

1993년 국방대학원 전산과학 석사

1997년~현재 충남대학교 전산과학과 박사과정

1995년~현재 육군 훈련 시뮬레이션 개발 책임

관심분야 : 분산/병렬시스템, 시뮬레이션, HLA

배 종 환

e-mail : mcguyver3515@msn.com

1997년 한남대학교 전자공학과 졸업

1998년~현재 제1정보통신 소프트웨어 개발부

관심분야 : 분산시뮬레이션, 객체지향프로그래밍, HLA

류 재 철

e-mail : jcryou@home.cnu.ac.kr

1985년 한양대학교 산업공학 학사

1988년 아이오와 주립대 전산학 석사

1990년 노스웨스턴 대학 전산학 박사

1991년~현재 충남대학교 컴퓨터과학과 교수

관심분야 : 컴퓨터 보안, 분산처리시스템