한국지능정보시스템학회
02 춘계학술대회 논문집, pp. 191~200

지능형 정보기술을 활용한 시뮬레이션/시뮬레이터 개발에 관한 연구 (HLA/RTI 中心)

A Study on the Simulation/Simulator Development using Intelligent Information Technologies* (HLA/RTI Oriented)

김화수**, 박영철***, 이경원***, 곽남선***

요 약

오늘날 발전을 거듭하고 있는 첨단 정보기술을 이용한 지능형 시뮬레이션/시뮬레이터의 표준으로 확고히 자리를 잡고 있는 HLA(High Level Architecture)는 시뮬레이션 소프트웨어의 재사용성과 상호운용성을 촉진시키기 위하여 미 국방성에 의해 개발되어 이제는 분산 시뮬레이션의 표준이라 할 수 있다.

이 논문에서는 지능정보 기술이 시뮬레이션/시뮬레이터 기술과 어떻게 연관되어 있으며, 어떻게 사용될 것인가에 대한 개념 연구와, 시뮬레이션/시뮬레이터모델 내에서 첨단 정보기술들이 무슨 역할을 하는지에 대한 방향을 도출하였다. 또한, HLA와 HLA의 가장 중요한 구성 요소중 하나인 RTI(Run Time Infrastructure)의 최신 버전과 RTI가 지원하는 서비스에 대해 고찰하였다. HLA 페더레이션 개발자들이 HLA 페더레이션 개발시 지침이 되는 여섯 단계의 페더레이션 개발 절차인 FEDEP(Federation Development and Execution Process) 모델을 따랐으며, RTI의 최신 버전(RTI-NG 1.3v4)을 사용하여 HLA를 따르는 지능정보형 시뮬레이션 프로그램의 프로토타입을 개발하였다.

1. 서 론

국내·외를 막론하고 민수용이든 군사 목적용이던 높은 기능과 성능이 요구되며, 매우 가격이 비싸고 복잡한 시스템, 특히 운용시 위험이 내재된 시스템에서는 예외 없이 각종 시뮬레이션/시뮬레이터들이 사용되고 있으며, 앞으로 이러한 시뮬레이션/시뮬레이터의 사용추세는 점차 확장될 것이다.

현대에 와서는 컴퓨터, 멀티미디어, 인공지능 기술 그리고 지능형 가상현실 분야가 급속히 발전하였고, 이러한 거의 모든 첨단 기술들이 시뮬레이션/시뮬레이터에 적용되어 발전되고 있다.

특히 컴퓨터 통신의 발달로 분산 시뮬레이션이 가능해졌으며, 1983년 SIMNET(SIMulator NETworking)에서부터 출발한 분산 시뮬레이션 기술은 발전을 거듭하여 현재는 분산 시뮬레이션을 위한 표준으로 자리잡은 HLA(High Level Architecture)로 이르고 있다.

그러나, 우리나라는 민수용이든 군사목적용이든 시뮬레이션/시뮬레이터의 중요성은 인식하고 있지만 이에 대하여 낮은 정책우선순위를 부여하고 있으며, 주요 국가 정보화/과학화 사업에서 등한시되고 있는 실정이다.

또한 현재는 외국기술에 일방적으로 의존하고 있으며 관련기술에 대한 인프라가 부족한 것도 문제점이라 할 수 있으며 이것은 디지털 혁명시대에서의 국가경쟁력제고에 걸림돌로 작용할 수 밖에 없을 것이다.

이에 본 논문에서는 새로운 지능형 정보기술을 활용한 시뮬레이션/시뮬레이터 개발 시 HLA 기반으로 개발하기 위한 기초/기반을 제공하고, 비 HLA 기반으로 운영되고 있는 시뮬레이션/시뮬레이터 모델을 HLA 기반으로 전환하여 분산 환경에서 사용자가 편리하게 활용할 수 있도록 한다는데 의의가 있다.

---

* 본 논문은 '01년도 (주)M&D 정보기술 용역과제 (2001. 3. 2)에 의거 일부 연구비를 지원 받아 연구를 수행하였음.

** 국방대학교 교수(전산정보학과)
***국방대학교 석사과정(전산정보학과)

이 논문에서는 가상현실 기술, 인간과 컴퓨터 상호작용(HCI), GIS 기술 등 첨단 정보기술을 이용한 지능형 시뮬레이션/시뮬레이터 개발에 관하여 연구하였으며 다음과 같이 구성되어 있다.

제 2장에서는 지능형 정보기술에 대하여 간략하게 살펴본다. 제 3장에서는 HLA 및 RTI의 등장 배경과 구성요소, 그리고 각 구성요소의 역할에 대해 기술한다.

제 4장에서는 RTI를 이용하여 HLA 페더레이션을 개발하고자 할 때 지침이 될 수 있는 FEDEP모델을 중심으로 기술한다.

제 5장에서는 HLA 페더레이션의 프로토타입 개발에 대해 기술하고, 제 6장에서는 결론 및 기대효과에 대하여 기술한다.

2. 관련 지능형 정보기술 조사 및 분석

2.1 가상현실 기술

가상현실이란 예술과 첨단기술이 만나는 곳, 현실에 구애됨이 없이 시간과 공간을 초월하여 상상의 세계가 현실과 같이 그대로 펼쳐지는 것, 즉, "인공 현실 감각"으로 정의할 수 있다.

또한 가상현실 기술은 컴퓨터 그래픽스, 멀티미디어, 대규모 병렬처리, 기가비트 정보통신 시스템 소프트웨어 기술, 분산처리 및 음성인식 기술 등의 기반이 꽃을 피울 수 있는 복합 기술로서 새로운 수요를 유발할 것이다.

이 가상현실 기술의 가장 효용성이 높은 분야가 바로 시뮬레이션/시뮬레이터 분야라 해도 과언이 아닐 것이다. 실제 현실의 문제를 컴퓨터 내에서 실제에 가깝도록 모형화 할 수 있는 3차원 가상현실 기술로 개발한 비행용 시뮬레이터, 함정용 시뮬레이터, 각종 군사 목적용 CBT 시뮬레이터 등의 예에서와 같이 가상 현실 기반의 시뮬레이션 모델 개발은 더욱 다양화되고 그 활용범위도 확대되어 가고 있다.

2.2 전문가 시스템

전문가시스템은 추론엔진 모듈, 지식베이스 모듈, 사용자 인터페이스 모델, 설명 모듈 등으로 구성된다.

이러한 전문가시스템은 정치, 경제, 군사, 문화, 과학기술, 의료 등 각종분야에 응용할 수 있으며 특히 국방부문에서는 각종 정보분석용, 교육훈련지원용, 정보처리지원용, 지휘관의 의사결정지원용 전문가시스템, 고가의 국방부문 특수 목적용 시스템 등 각종 시스템의 고장진단 및 조치를 수행하는 전문가 시스템 등에 다양하게 응용되고 있다.

또한 전문가시스템은 시뮬레이션/시뮬레이터 모델 개발 시에도 다양하게 활용 가능하다.

예를 들어, 미국의 CBS 모델을 바탕으로 한국형 워게임 모델로 개발한 '창조 21' 워게임 모델에는 전문가 시스템의 주요 모듈 중의 하나인 설명 모듈을 적용하여 청·홍군 간에 전투에 관련된 각종 정보를 분석하는 사후분석 기능을 수행하였다.

2.3 HCI 기술

그래픽 사용자 인터페이스(GUI : Graphic User Interface)에 대한 연구와 컴퓨터 하드웨어의 가격 폭락이 컴퓨터의 대중화에 큰 역할을 했으며, 이를 구현할 수 있는 기술적인 발전과 함께 인간이 어떻게 생각하고 행동하는 지에 대한 연구들이 접목되면서, 컴퓨터와 인간의 상호 작용 방법들이 급진전되었다.

대부분의 시스템 개발자들은 얼마나 그 시스템을 다른 기술자들이 쉽게 제어할 수 있고 얼마나 효율적으로 컴퓨터에서 돌아갈 수 있느냐에 초점을 맞추어 시스템을 설계하고 있다. 하지만 HCI는 사용자들이 얼마나 시스템을 편리하고 안전하게 사용할 수 있느냐 하는 측면을 강조하고 있는 만큼 시뮬레이션/시뮬레이터 개발 시에도 사람으로 하여금 보다 친밀감과 편안함을 느끼고, 기계라는 거부감을 느끼지 않도록 고려하여야 하므로 HCI 기술은 시뮬레이션/시뮬레이터를 개발하는데 중요한 기술 중의 하나이다.

2.4 GIS 기술

GIS(Geographic Information System)는 자연 및 사회, 경제적 정보를 지리적 공간위치에 맞추어 입력, 저장해서 여러 목적에 맞게 활용, 분석하는 기술로써 각종 데이터의 수집과 처리작업에 대해 경제성과 능률성을 제공해 주며 디지털 컴퓨터의 이용으로 데이터 저장 및 공간정보 이용에 획기적인 계기를 마련해 주었다.

또한, GIS는 실제 세계의 모델을 의미하며 자료는 상호관련 되어 접근, 변화, 관리되기 때문에 환경변화의 분석, 경향분석 또는 의사결정과 결과의 예측 등을 실험할 수 있는 모델이다.

이는 컴퓨터 기술과 공간자료를 효율적으로 이용하고자 시작되었는데, 자료수집 방법은 종래의 지도나 보고서, 최근에는 인공위성이나 항공촬영용 비행기로부터 센서를 통하여 입소된 HDDT(High Density Digital Tape)의 수집 방법이 중요하게 작용하고 있다.

이러한 GIS 기술을 이용하여 시뮬레이션/시뮬레이터 객체들의 위치정보를 GIS DB로 구축하고 객체들의 디지털 위치정보를 시뮬레이션/시뮬레이터 모델 개발 시 합성 공간에 활용함으로써 지능형 GIS 기술이 시뮬레이션/시뮬레이터 기술발전에 많은 도움을 줄 것으로 기대되며, 이외에도 시뮬레이션/시뮬레이터에 관련된 핵심적인 기술이 많으나 지면관계상 생략하도록 하겠다.

3. HLA 및 RTI 일반

3.1 HLA의 등장 배경 및 구성요소

3.1.1 HLA 배경

미 국방성이 분산 시뮬레이션을 위해서 HLA와 같은 고 수준의 구조를 개발하게 된 배경을 살펴보면 다음과 같다.

첫째, 미 국방예산의 삭감을 들 수 있다. 야전훈련과 같은 막대한 예산이 소모되는 대규모 군사 훈련은 비용의 측면 때문에 많은 제한을 받게 되었으며, 이에 대한 가장 효과적인 해결책으로 제시된 것이 시뮬레이션/시뮬레이터의 활용이었고, 특히 HLA와 같은 고 수준의 개념이 적용된 시뮬레이션/시뮬레이터 기술을 무기체계의 획득에서 운영, 각종 군사훈련 그리고 후속지원과 연구개발의 전 단계에 적용함으로써 비용절감을 이루겠다는 것이다.

둘째, 더욱 복잡해져 가는 군사 작전상의 요구사항들이다. 다양한 전장 환경 하에서 다양한 형태의 작전을 수행하기 위해서는 전통적인 훈련방식만으로는 그 요구사항을 충족할 수가 없었으며, 이러한 상황 하에서 시뮬레이션/시뮬레이터를 이용한 다양한 형태의 훈련이 그 대안으로 제시되었고, 상호운용성에 기반을 두고 개발된 HLA가 그 해법이 될 수 있었다.

3.1.2 HLA의 구성 요소

가. HLA 규칙

HLA 규칙은 페더레이션에 있는 시뮬레이션의 적절한 상호작용을 보장하고, 시뮬레이션과 페더레이트의 책임을 기술한 것이다. 여기서 적절한 상호작용을 보장한다고 하는 것은, 후자의 '책임'과 연결되는 의미로써, HLA를 따른다고 여겨지기 위해서는 HLA Rule을 지켜야 하며, 이 Rule을 지키게 되면 자연히 적절한 상호작용이 보장되는 것이다.

즉, 최상위 수준에서, HLA는 만약 페더레이트 또는 페더레이션이 HLA를 따른다고 여겨지기 위해서 준수해야 하는 10개의 HLA 규칙들을 정의하였다. HLA 규칙은 HLA 페더레이션을 위한 5개의 규칙(Federation Rules)과 HLA 페더레이트를 위한 5개의 규칙(Federate Rules), 두 개의 그룹으로 나뉜다.

- 규칙 1은 문서요구 사항을 규정한 것이다. 즉 페더레이션마다 각자 자신의 FOM(Federation Object Model)을 가져야 한다는 것으로, 페더레이션을 개발하고자 하는 개발자는 개발의 어느 단계(주로 페더레이션 설계가 끝나고 실제 개발에 들어갈 때)에서는 자신이 개발하고자 하는 페더레이션을 위한 FOM을 생성해야 함을 의미한다.
- 규칙 2는 객체의 표현에 관한 것으로, 자신이 개발하고자 하는 페더레이션의 목표를 달성하기 위해 생성되는 모든 객체는 RTI가 아닌 자신의 페더레이트 내에 존재해야 함을 규정한 것이다.
- 규칙 3은 자료교환에 관한 것으로, 모든 자료가 RTI를 통하여 교환되게 함으로써 자료를 효율적으로(자료교환과 관련된 통신양의 측면에서) 상호 교환할 수 있게 된다.
- 규칙 4는 인터페이스 요구사항에 관한 것이다. 개발되는 페더레이션이 서로 동일한 인터페이스를 사용하게 됨으로 해서 상호운용성을 증진시킬 수 있다.
- 규칙 5는 소유권속성에 관한 것이다. 객체 인스턴스의 속성에 대한 소유권을 오직 하나의 페더레이트가 갖도록 하고, 또한 객체에 대한 소유권을 두 개 이상의 페더레이트가 공유하게 될 때 생성과 소멸에 대한 접근을 상호 배타적으로 가능하게 함으로써, 객체 인스턴스의 생성과 소멸에 대한 효과적인 관리를 제공한다.

  이때 주어진 객체 인스턴스를 소멸 시킬 수 있는 권리는 해당 인스턴스를 소유한 페더레이트로 한정된다.
- 규칙 6에서 10은 페더레이션에 참가하는 개별적인 페더레이트에 관련된 것들을 다룬다. 규칙 6은 규칙1과 마찬가지로 문서요구사항을 나타내며, 페더레이션이 하나의 FOM을 가져야 하듯이, 페더레이트는 하나의 SOM (Simulation Object Model)을 가져야 한다.
- 규칙 7, 8 그리고 9는 관련된 객체 속성의 제어와 전송을 위한 규칙이며, 규칙10은 시간관리에 대한 것이다.

나. 인터페이스 명세(Interface Specification)

인터페이스 명세는 RTI를 위한 표준을 정의한다. 즉, RTI의 서비스에 대한 정의가 인터페이스 명세라고 할 수 있다.

인터페이스 명세는 페더레이트가 어떻게 페더레이션과 상호작용을 하고, 궁극적으로 페더레이션 내의 다른 페더레이트들과 어떻게 상호작용을 수행하는지 정의한다.

다. 객체모델 템플릿(Object Model Template)

OMT는 페더레이션 객체 모델(FOM), 시뮬레이션 객체모델(SOM), 그리고 관리 객체모델(MOM)을 정의한다.

페더레이션 객체모델(FOM)은 페더레이션마다 하나씩 존재하는 것으로 객체와 그들 객체간의 상호작용등 모든 공유된 정보를 기술하며, 이러한 정보들은 페더레이션간의 문제에 초점이 맞추어진 것들이다.

FOM의 주된 기능은 페더레이트간의 데이터 교환의 특징을 일반적인 표준화된 포맷으로 명시하는 것이다.

시뮬레이션 객체모델(SOM)은 페더레이트마다 하나씩 가진다. 이것은 주로 페더레이트의 특징을 기술하며, 외부적으로 사용될 수 있는 객체와 상호작용에 대해 기술하며, 페더레이션 객체모델과 달리, 페더레이트의 내부 작동에 초점이 맞추어 진다.

관리 객체모델은 그 명칭에서 유추할 수 있듯이, 페더레이션 관리에 사용되는 객체와 상호작용에 대해 기술하는 것이다.

HLA에서는 객체모델을 문서화하기 위해 이러한 공통 구조를 제공함으로써 궁극적으로 시뮬레이션과 그 구성요소들의 상호운용성과 재사용성을 촉진시킬 수 있다. 다음의 7가지 구성요소는 OMT 명세서에 의해 요구되는 것들이다.

1) 객체모델 식별 테이블
(Object Model Identification Table)

객체모델 식별 테이블의 용도는 페더레이션 또는 페더레이트에 대한 식별정보 키를 제공하는 것이다.

2) 객체 클래스 구조 테이블
(Object Model Identification Table)

HLA 객체 모델의 객체 클래스 구조는 시뮬레이션 또는 페더레이션 도메인에 존재하는 객체의 클래스들 간의 관계의 집합으로 정의되어야 한다.

HLA 객체 클래스는 독특한 특성 또는 속성을 공통으로 가지는 객체들의 모임이며 클래스의 각각의 개별적인 객체는 그 클래스의 인스턴스이다.

3) 상호작용 클래스 구조 테이블 (Interaction Class Structure Table)

한 페더레이트에 의해 수행되어 있는 다른 페더레이트내의 객체에 형향을 줄 수 있는 활동을 표현하는 테이블이다.

4) 속성 테이블 (Attribute Table)

FOM의 속성 테이블은 페더레이션 내에 표현된 모든 객체 속성에 대한 정보를 서술적으로 제공하도록 설계되었으며, 테이블 구조는 여러 개의 열로 구분된다.

5) 파라미터 테이블 (Parameter Table)

HLA 객체 모델의 파라미터 테이블은 페더레이션 내에 표현된 모든 상호작용의 모든 파라미터에 대한 설명적인 정보를 제공하기 위하여 설계되었으며, 테이블 구조도 속성테이블과 마찬가지로 여러 개의 열로 구분된다.

6) 라우팅 공간 테이블 (Routing space Table)

테이블에서 최상위 레벨에 기록되어야 하는 것은, 첫 번째 열에 제공되는 라우팅 공간이다. 그것의 이름은 라우팅 공간에 의해 필터링 되는 형태를 나타내는 것이 사용된다. 다음 열은 라우팅 공간 범위의 이름을 기입한다. 각 라우팅 공간은 하나 또는 그 이상의 범위를 가질 수가 있고, 각각은 라우팅 공간의 구체적인 특성을 나타낸다.

7) FOM/SOM 사전(FOM/SOM Lexicon)

FOM/SOM 사전은 FOM을 작성하는 동안 사용되는 모든 항목의 정의를 문서화하는 수단을 페더레이션에 제공하며, 개별적인 페더레이트가 그들의 SOM에서 모든 항목의 정의를 문서화하는 수단을 제공한다.

3.2 RTI의 구성요소 및 실행 개요

3.2.1 RTI 구성요소

가. rtiexec(RTI Executive Process)

rtiexec는 다수의 페더레이션 실행의 생성과 소멸을 관리해주는, 하나의 플랫폼 상에서 실행되는 전역 프로세스이다. 이것은 각 fedex가 서로 다른 이름을 갖도록 보장해주고 fedex와 마찬가지로 수동 조작을 위한 console 인터페이스를 제공한다.

나. fedex(Federation Executive Process)

fedex는 실행중인 페더레이션마다 실행되는 하나의 프로세서이며, 페더레이션에 첫 번째로 참가한 페더레이트에 의해 생성된다. 각 fedex는 하나의 페더레이션을 관리하고, 페더레이션에 참가하는 각 페더레이트에 고유의 핸들을 할당하며, 다수의 페더레이트가 페더레이션 실행에 참가하고 탈퇴하는 것을 관리해준다.

또한 fedex는 페더레이트 간의 자료교환을 용이하게 해주고, 수동 조작을 위한 콘솔 인터페이스를 제공한다. 즉, 페더레이션 실행 중에, 현재 페더레이션에 참가하고 있는 페더레이트의 현황 등과 같은 정보를 fedex를 통하여 모니터링 할 수 있다.

다. libRTI(RTI Library)

C++ 라이브러리인 libRTI는 페더레이트 개발자에 게 HLA 인터페이스 명세서에 명시된 서비스들을 제공해 준다. 페더레이트들은 HLA 서비스를 호출하기 위해 rtiexec, fedex 그리고 다른 페더레이트들과 통신하는 libRTI를 사용한다.

HLA 인터페이스 명세서는 libRTI가 페더레이트에게 제공하는 서비스와 각 페더레이트가 페더레이션에게 갖고 있는 책임을 정의한다. libRTI 내에서 RTIambassador 클래스는 RTI에 의해 제공되는 서비스들을 구현하고 있다.

RTI에 대해 페더레이트가 만든 모든 요구는 RTIambassador의 메서드 형태로 호출된다. 추상화 클래스인 FederateAmbassador는 각 페더레이트가 제공해야 하는 callback 함수를 알려준다.

3.2.2 RTI를 이용한 페더레이션 실행 개요

이 절에서는 RTI의 여러 구성요소들이 실제 페더레이션이 실행될 때 어떠한 역할을 하게 되며, 어떻게 상호작용 하는지를 기술한다.

<그림 1>은 페더레이션 실행에 대한 각 프로그램과 페더레이트 그리고 FED파일과 RID파일간의 상호작용을 도시한 것이다.

위의 시스템에서는 두 개의 페더레이션이 실행중이다. 비록 하나의 시스템에 두 개의 페더레이션이 실행된다 하더라도, 그것들은 서로 독립적이며 어떠한 정보도 교환하지 않는다. 이 시스템이 가지는 각각의 구성요소는 다음과 같다.

첫째, 양쪽 페더레이션에 의해 공유되는 RTIExec 그리고 RTI.RID 파일이 있다. RTIExec 프로그램은 앞에서 설명했듯이, 다수의 페더레이션의 생성과 소멸을 관리하는 프로그램으로써, 두 개의 각 FedExec 프로그램(FedExec1과 FedExec2)이 고유의 이름을 갖도록 보장해준다.

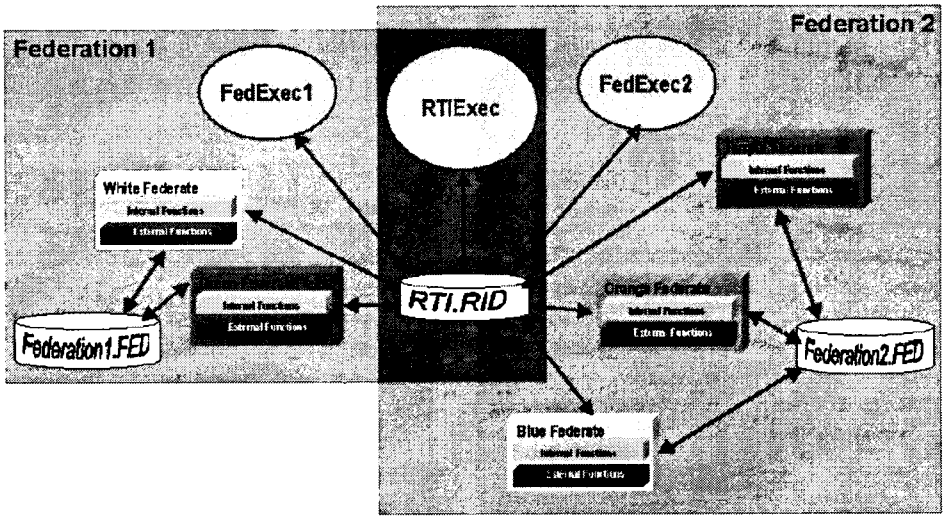
둘째, 두 개의 FedExec 프로그램이 있다. FedExec1은 페더레이션1의 실행을 통제하고, FedExec2는 페더레이션2의 실행을 통제한다. 위에서 기술한 바와 같이 페더레이션1과 페더레이션2는 서로 어떠한 정보도 교환하지 않는 별개의 것이다.

셋째, 페더레이션 1에는 white, green 두 개의 페더레이트가 있고, 페더레이션 2에는 purple, orange, blue 세 개의 페더레이트가 있다.

이러한 기본적인 페더레이션 실행의 구성요소들이 서로 어떠한 절차를 거쳐서 실행되는가를 상기 그림의 페더레이션 1의 측면에서 살펴보면 다음과 같다.

먼저, 페더레이션이 실행되기 이전에 사용자는 RTI를 시작하여야 한다. 이것은 간단히 RTIExec 프로그램을 실행시키는 단순한 작업이다. RTIExec가 작동되는 상태에서 페더레이트는 마치 관리자와 같이 작동하게 되는데, 사용자가 페더레이트를 실행시키면 즉, 상기의 white(또는 green) 페더레이트를 실행시키면 페더레이트는 자신의 RTIAmbassador 상의 createFederationExecution() RTI 메서드를 호출하여 페더레이션을 생성한다.

그 다음 RTIAmbassador는 RTIExec에 페더레이션 실행의 이름(FedExec1)을 저장하고, 또한 자신의 통신 주소를 FedExec에 등록한다. 페더레이션 실행은 계속 진행중이다.



<그림 1> RTI를 이용한 페더레이션 실행

하나의 페더레이션 실행(FedExec1)이 존재하면, 다른 페더레이트(green 페더레이트)는 그곳에 참가할 수 있다. RTIAmbassador는 FedExec의 주소를 알기 위해 RTIExec를 참조하고 FedExec상에서 joinFederationExecution() 메서드를 호출하여 페더레이션에 참가한다. 다른 페더레이트들도 동일한 절차를 거쳐서 페더레이션에 참가할 수 있다.

이외에도 RTI의 중요한 역할은 RTI의 지원서비스가 있다. RTI의 지원서비스에는 페더레이션 관리, 선언 관리, 객체 관리, 소유권 관리, 시간 관리, 자료분배 관리가 있으며, 이에 대한 상세한 내용은 관련 참고 문헌을 참조 바란다.

4. HLA 페더레이션 개발 절차

FEDEP(Federation Development and Execution Process) 모델의 목적은 페더레이션 개발자들이 그들의 애플리케이션 요구를 달성하는 수단이 되는 페더레이션 개발과 실행을 위한 지침을 제공하며, 각 단계를 요약하면 다음과 같다.

- 단계 1은 페더레이션 목적을 정의하는 것으로 첫 번째는 요구사항 식별 단계로 이 활동의 주요한 목적은 페더레이션에 의해 다루어지는 문제들의 명확한 이해를 돕는 것이며, 이 활동의 결과로 산출되는 요구설명서(Needs Statement)는 그 범위와 형식화의 정도에 있어서 매우 광범위하다. 두 번째는 목표 개발 단계로 이 활동의 목적은 요구설명서를 페더레이션을 위한 더 상세하고 명확한 목표들로 바꾸어 주는 것이다.
- 단계 2는 페더레이션 개념 모델을 개발하는 것으로 첫 번째는 시나리오 개발 단계로 페더레이션 시나리오의 기능적인 명세를 개발하는 것이다. 두 번째는 개념 분석 수행 단계로 개념 분석 활동 중에 페더레이션 개발팀은 사용자 요구사항 판단, 페더레이션의 목표, 그리고 정의된 환경에 기반을 둔 계획된 문제 영역의 개념적인 표현을 개발한다. 세 번째는 페더레이션 요구사항을 개발하는 단계로 페더레이션 요구사항은 페더레이션 목적에 기초한 것이며, 즉시 시험가능 해야 하며 페더레이션을 설계하고 개발하는데 필요한 실행 수준의 지침을 제공해야 한다.
- 단계 3은 페더레이션을 설계하는 단계로, 첫 번째는 페더레이트를 선택하는 것이다. 이 단계에서 해야 하는 활동은 장차 페더레이션의 참가자가 되는 개별적인 시뮬레이션 시스템의 적합성을 결정하여 적절한 페더레이트를 선택하는 것이다. 두 번째는 기능을 할당하는 것으로 한번 모든 페더레이트가 식별되면, 그 다음의 주요한 활동은 페더레이션 개념 모델 내에 있는 엔티티와 행동들을 재연하기 위한 책임을 페더레이트에게 할당하는 것이다. 세 번째는 계획을 준비하는 것으로, 수행하게 되는 작업은 페더레이션의 개발, 시험, 그리고 실행을 관리하기 위한 통합된 계획을 개발하는 것이다.
- 단계 4는 페더레이션을 구현하는 것으로, 첫 번째는 FOM 개발이다. FOM을 개발하기 위해서는 이전에 페더레이션의 한 부분으로 식별된 각각의 페더레이트가 사용되며, 개발시 객체모델 개발 툴(OMDT : Object Model Development Tool)과 같은 자동화 도구의 사용이 강력히 추천된다. 두 번째는 페더레이션의 합의를 수립하는 것으로 FOM내에 문서화될 필요가 없는 페더레이트들 사이에 도달해야 하는 다른 형태의 합의는 충분히 일관된 상호운용 가능한 분산 시뮬레이션 환경의 수립에 필수적이다. 세 번째는 페더레이션 수정 실행하는 것으로 이 활동의 목적은 페더레이션 개념 모델에 묘사된 것처럼 할당된 객체와 관련된 행동을 표현 할 수 있음을 페더레이트에게 보장하기 위하여 수정이 필요한 것은 무엇이든 구현하는 것이다.
- 단계 5는 페더레이션 통합과 시험을 하는 것으로 첫 번째는 실행 계획을 수립하는 것이

다. 이 활동의 목적은 HLA 페더레이션을 지원하기 위해 요구되는 모든 정보를 정의하고 개발하는 것이다. 두 번째는 페더레이션 통합으로 이 활동의 목적은 모든 페더레이션 참가자들을 하나의 통합된 운영환경 내로 가져오는 것이다. 세 번째는 페더레이션 시험으로 이 단계에서는 모든 페더레이션 참가자들이 페더레이션 목표를 달성하기 위해 요구되는, 상호운용이 가능한가를 시험하게 된다.

- 단계 6은 페더레이션 실행과 결과를 비교하는 것으로, 첫 번째는 페더레이션 실행이다. 이 단계에서는 모든 페더레이션 참가자들이 요구되는 출력물을 생성하기 위하여 전체적으로 통합된 하나로써 실행되는 것이며, 정해진 페더레이션의 목표를 성취하는 것이다. 두 번째는 출력을 처리하는 것으로 이 활동의 목적은 페더레이션 실행 동안에 수집된 출력물에 대한 후기의 처리 작업이다. 세 번째는 결과를 분석하는 것으로, 이 활동은 추론된 결과들이 모든 페더레이션 목표를 충족했는지를 평가하고, 재사용 가능한 모든 페더레이션 산출물들을 적절한 장소에 저장하는 것이다.

5. HLA 페더레이션 프로토타입 개발

5.1 프로토타입 개발 개요

5.1.1 구현 환경

개발하고자 하는 프로토타입은, 프로그래밍 언어로는 Visual C++ 6.0과 SP 5.0을 사용하여 개발하며, RTI는 2001년 6월 29일에 릴리즈된 최신 버전인 RTI-NG 1.3v4를 사용하였다.

객체모델의 개발을 위해서는 객체모델 개발 도구인 OMDT 버전 1.3v5를 사용한다. 작동 환경은 Windows 98 O/S의 PC를 사용하는 시스템이며, 동일한 네트웍 상에서 분산되어 작동하는 것으로 한다.

개발하려고 하는 프로토타입은 두 개의 페더레이트로 구성된 페더레이션이며, 두 개의 페더레이트는 각각 windows98 O/S를 사용하는 PC상에서 작동되며, 서로 동일한 네트웍으로 연결되어 있다. 하나의 PC에서는 RtiExec 프로그램과 하나의 페더레이트(Federate 1), 그리고 FedExec가 실행되고, 다른 하나의 PC에서는 두 번째 페더레이트(Federate 2)가 실행된다.

5.1.2 프로토타입의 목표

프로토타입 개발의 목적은 오로지 연구를 위한 것이다. RTI의 여러 가지 서비스들을 사용하여 실제 페더레이션을 개발함으로써 서비스들의 구현과 페더레이션 개발의 절차를 제시하고자 하는 것이다.

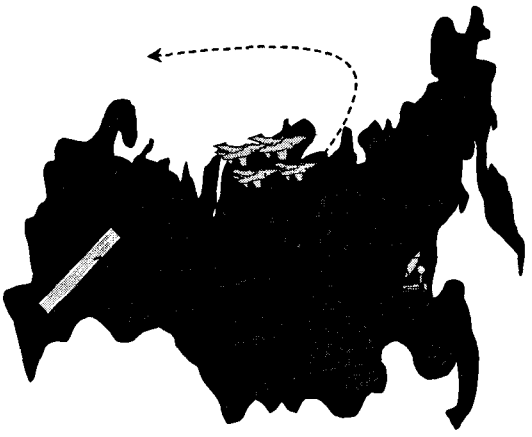
이러한 목적 하에 개발하려고 하는 프로토타입에 관한 주요 요구사항을 페더레이션 사용자/발주자의 측면에서 기술하면 다음과 같다.

첫째, 페더레이션 내에 두 개 이상의 페더레이트가 실행되어야 한다. 둘째, 페더레이션 개발 시 이전에 개발된 가용 자원(예를 들어, FOM, SOM 등)은 사용하지 않고 처음부터 완전히 새롭게 개발한다. 셋째, 각 페더레이트는 분산된 서로 다른 플랫폼 상에서 실행되어야 한다. 넷째, 기타 사용 가능한 자원(예를 들어, 자금, 인원, 시설 등)이나 개발에 영향을 미치게 되는 제약사항(예를 들어, 개발기간, 보안 요구사항 등)은 없는 것으로 가정한다.

5.2 페더레이션 시나리오

5.2.1 시나리오 개발

개발하고자 하는 프로토타입의 시나리오가 <그림 2>에 나타나 있다.



<그림 2> 페더레이션 시나리오

- 초기 조건으로써 Air_Federate는 4대의 항공전력을 가지며, Ground_Federate는 지상군 1000명과 대공 미사일 전력을 가진다.
- 항공기 전력은 정해진 경로를 따라 적 지역으로 비행을 시작한다.
- 사용자의 조작에 의해 적 지역에 폭탄을 투하한다.
- 지상의 대공 미사일 전력은 항공기 침투 시 사용자의 조작에 의해 항공기를 향해 대공 미사일을 발사한다.
- 교전후의 전력 손실에 대해 화면상에 표시해 준다.
- 한번의 비행으로 모의가 끝나며, 사용자의 조작에 의해 동일한 시나리오로 재시작 할 수 있다.

5.2.2 개념 분석 수행

개발하고자 하는 프로토타입의 핵심적인 객체클래스와 관련된 속성에 대한 사항이 <표 1>에 나타나 있으며, 식별된 상호작용은 <표 2>와 같다.

<표 1> 프로토타입의 객체클래스와 관련 속성

Federate	Class	Attribute
Air_Federate	Airplane	Location
		State
		Remains
	Bomb	Location
Ground_Federate	Troops	State
		Remains
	Sam	Location

<표 2> 프로토타입의 상호작용

상호작용	내 용
FireBomb	항공기 객체에서 지상을 향하여 Bomb을 투하한다.
BurstBomb	투하한 Bomb이 폭발한다. 파라미터로 BombTarget을 넘겨준다.
FireSam	침투항공기에 대공 미사일을 발사한다.
BurstSam	발사한 대공 미사일이 폭발한다. 파라미터로 SamTarget을 넘겨준다.
Reset	페더레이션을 페더레이트들이 처음 Join 했을 때의 초기 상태로 돌려준다.

5.3 페더레이션 설계 및 구현

개발하고자 하는 두 개의 페더레이트는, 하나는 지상을 공격하는 항공기 전력을 모의하고, 다른 하나는 공격 항공기에 대한 대공 방어를 수행하는 역할을 모의한다. 편의상 지상 공격을 수행하는 페더레이트를 Air_Federate 라고 하고 아군이라 가정하며, 대공방어를 수행하는 페더레이트를 Ground_Federate 라고 하고 적이라 가정한다.

각각의 페더레이트에 할당되는 기능은 <표 3>과 같다.

5.3.1 페더레이션 구현

가. FOM 개발

프로토타입 페더레이션을 위한 FOM을 개발하기 위하여 객체모델 개발도구(OMDT : Object Model Development Tool)를 사용하였다.

사용된 OMDT는 버전 1.3v5 이며, 6.3.2절의 페더레이션 시나리오와 개념분석을 바탕으로 Battle 페더레이션을 위한 FOM과 두 개의 페더레이트를 위한 각각의 SOM을 개발하였으며, 개발된 FOM과 OMDT를 이용하여 Battle.FED 파일을 생성하였다.

<표 3> 각 페더레이트에 할당된 기능

페더레이트	기 능 할 당
Air_Federate	초기 항공 전력 묘사(4대)
	정해진 비행경로를 따라 비행함. 이때 비행경로는 적 진영을 통과하도록 해야 한다.
	적 진영에 폭탄 투하
	폭탄 투하후의 적 전력 상태 표시
	페더레이션 실행 후 LOG 출력
Ground_Federate	초기의 지상 전력 묘사(1000명)
	항공기 침투시 대공 미사일 발사
	대공 미사일 발사후의 적 항공전력 상태 표시
	페더레이션 실행 후 LOG 출력

나. 프로토타입 구현

실제 구현한 프로토타입은 <표 4>에 나타난 것과 같이 13개의 소스 파일과 13개의 헤더 파일로 구성되어 있다.

<표 4> 프로토타입 구성 소스 파일

Source Files	Header Files
- ArtFederate-Ambassador.cpp - Battle.cpp - Battle.rc - BattleDoc.cpp - BattleView.cpp - byte_swap.cpp - CISBitmap.cpp - Log.cpp - LogViewDlg.cpp - MainFrm.cpp - Objects.cpp - rti_interface.cpp - StdAfx.cpp	- ArtFederate-Ambassador.h - Battle.h - BattleDoc.h - BattleView.h - byte_swap.h - CISBitmap.h - Log.h - LogViewDlg.h - MainFrm.h - Objects.h - Resource.h - rti_interface.h - StdAfx.h

각각의 파일들에 대한 간략한 설명은 다음과 같다.

- Battle, BattleDoc, BattleView, MainFrm은 윈도우 프로그램의 기본적인 구조를 만든다. 윈도우 창이나, 메시지 루프나 프레임 윈도우 등이다.
- 일단 BattleView가 윈도우와 연결된 처리를 하고(항공기를 그린다든가, 에디트 창에 숫자를 표시하거나, 버튼이 눌러졌을 때 등) 있다.
- CISBitmap은 비트맵 이미지에서 선택한 색상

을 배경에 투명하게 그려준다. 페더레이트에 만들어지는 항공기 객체나 또는 지상 전력에 대한 객체의 이미지가 투명한 배경을 가지도록 BattleView의 OnDraw에서 DrawParent()를 호출한다.

- byte_swap은 RTI에 파라미터를 전달하거나 받을 때, 바이너리를 네트워크 order와 로컬 호스트 order로 서로 바꿔주는 함수들을 가지고 있다.
  예를 들어, 윈도우 O/S에서는 long 변수를 비트 형태로 바꿔 봤을 때 1100이라고 한다면, 네트워크 order에선 0011이 된다. RTI가 네트워크 order를 사용하고 있기 때문에, 파라미터를 주고받기 전에 비트 순서를 바꿔주어야 하는 것이다.
- Log는 말 그대로 로그들을 저장하고, LogViewDlg는 로그 버튼을 눌렀을 때, 저장된 로그들을 보여주는 대화상자를 나타낸다.
- rti_interface는 RTI에 이벤트를 보내거나, 객체를 만들거나, 클래스 객체의 속성이 변경되었거나, 페더레이션에 참가하거나 탈퇴하기 위한 join, resign등을 하는 함수를 갖고 있다. 이러한 함수를 버튼이 눌러졌을 때 호출하기 위해, BattleView에서 rti_interface.h를 선언하고, 함수를 호출하게 된다.
- ArtFederateAmbassador는 RTI에 join한 뒤, 네트웍에 연결된 다른 프로그램에서 위에서와 같이 속성 변경 등의 RTI 함수를 호출했을 때, 현재 프로그램에 알려주는 함수들을 가지고 있다. 즉, 이벤트가 발생하면, RTI가 ArtFederate-Ambassador에 있는 함수들을 호출해서, 한쪽 프로그램에서 다른 쪽 프로그램의 이벤트가 발생했음을 알 수 있게 된다.
- objects는 배열형태로 네트웍에 연결된 다른 프로그램에서 만든 객체들을 관리하는 자료구조를 가지고 있다.

현재, 다른 쪽에서 객체를 만들면 ArtFederate-Ambassador에서 discoverObjectInstance()가 호출되고, objects 테이블에 객체 정보를 추가하는 식으로 되어 있다.

속성 변경 이벤트가 발생했을 때도 마찬가지로 objects에서 해당 객체를 찾아서 속성 값을 반영해준다. BattleView에서 다른 쪽 객체가 있는지, 혹은 속성 값을 참조하기 위해, #include "objects.h"를 삽입 해주고, objects에 대한 함수들을 호출해서 값을 얻어오게 된다.

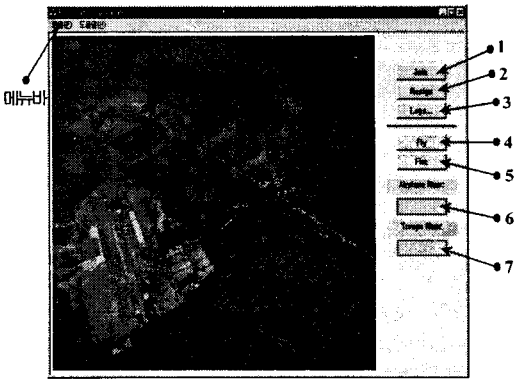
5.3.2 페더레이션 실행과 결과 분석

가. 페더레이션 실행

프로토타입을 실행한 화면이 <그림 3>에 나타나 있다.

모의하고자 하는 가상 전장상황은 '전장모의영역'에서 이루어지며, 프로토타입과 관련된 각 부분의 기능은 아래와 같다.

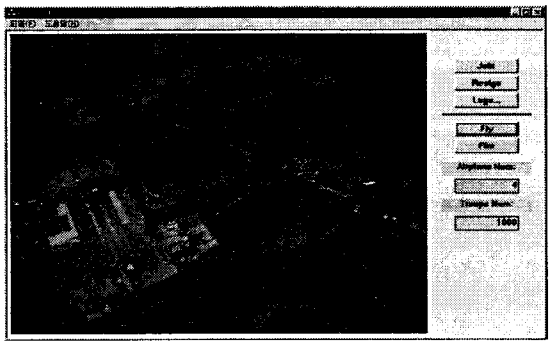
1. Join 버튼 : 페더레이션에 참가하기 위한 메뉴이며, click 하면 RTI의 joinFederation-Execution() 함수를 호출하여 페더레이션에 참가한다.
2. Resign 버튼 : 페더레이션 실행에서 탈퇴하기 위한 메뉴이다.



<그림 3> 프로토타입 실행 초기화면

3. Logs 버튼 : 페더레이션 실행 동안의 Log를 보여주는 메뉴이다.
4. Fly 버튼 : Air_Federate의 항공전력이 적 지상군을 공격하기 위한 비행을 시작하는 메뉴.
5. Fire 버튼 : Air_Federate의 경우 항공기에서 지상으로 Bomb을 투하하고, Ground_Federate의 경우 항공기를 향하여 대공 Missile을 발사한다.
6. 에디터 박스 : 항공전력의 상태를 표시해줌.
7. 에디터 박스 : 지상 전력의 상태를 표시해줌.

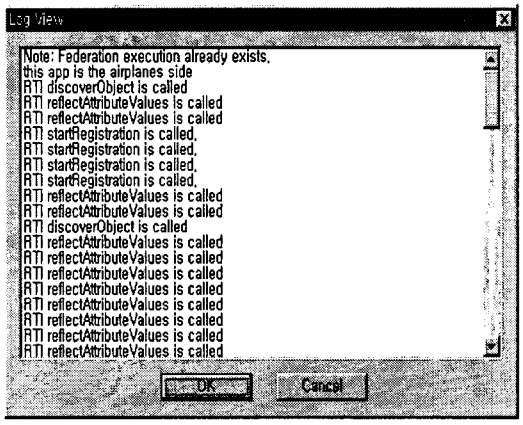
<그림 4>는 Air_Federate 그리고 Ground_Federate 두개의 페더레이트가 페더레이션에 참가하여 전장모의영역에서 객체를 생성하고 상호작용하는 것을 나타낸다.



<그림 4> 두 페더레이트의 페더레이션 참가

나. 결과 분석

페더레이션 실행 후에 Air_Federate 측면의 Log를 출력한 것이 <그림 5>에 나타나 있다. 그림에서 보여지는 바와 같이 페더레이션 실행과 객체의 발견, 객체의 등록과 발견, 객체 속성의 반영 등 일련의 RTI 서비스들이 이상 없이 수행됨을 알 수 있다.



<그림 5> 페더레이션 실행후의 LOG 출력

6. 결론 및 기대효과

6.1 결 론

현재의 시뮬레이션/시뮬레이터 관련 기술들은 여러 관련 기술들의 발전과 더불어 비약적으로 발전해가고 있다. HLA는 이미 분산 시뮬레이션 분야에서 그 위상을 확고히 하였으며, 또한 계속적인 발전을 거듭하고 있다. 이제 HLA는 시뮬레이션/시뮬레이터 분야의 거스를 수 없는 대세이며, 상호운용성과 재사용성 측면을 고려해 볼 때 우리 민수 및 국방분야에도 시급히 도입해야 할 중요한 기술이요 표준이다.

따라서 이 논문에서는 HLA와 HLA의 핵심이라고도 할 수 있는 RTI 그리고 HLA 페더레이션 개발 절차에 대해 연구하였으며, 이를 바탕으로 HLA를 기반으로 한 페더레이션의 프로토타입을 구현해 보았다. 차후 본 연구 결과가, 실제의 훈련에 사용될 수 있는 대규모의 페더레이션을 개발하는 기초가 될 수 있을 것이며, 또한 더 심도 깊은 연구를 통하여 RTI의 각종 서비스들을 개선할 수 있는 단계로까지 발전할 수 있는 출발점이 될 수 있을 것으로 기대된다.

6.2 기대효과

이 논문에서 연구된 내용은 크게 두 가지 측면에서 그 기대효과를 고려해 볼 수 있을 것이다.

첫째, 새로운 지능형 정보기술을 활용한 시뮬레이션/시뮬레이터 개발 시 HLA 기반으로 개발하기 위한 기초/기반을 제공한다.

HLA를 도입하려하는 초기 단계에 있는 상황하에서, HLA 기반의 새로운 시뮬레이션/시뮬레이터 개발에 대한 기초 연구를 제공한다.

둘째, 현재 민수용, 군사목적용으로 운영하고 있는 비 HLA 기반의 시뮬레이션/시뮬레이터 모델을 HLA 기반으로 전환하여 분산 환경에서 사용자가 편리하게 실행할 수 있도록 인공지능기술, 가상현실 기술, HCI기술 등 최신의 지능정보기술을 활용한 시뮬레이션/시뮬레이터를 개발하는데 참고가 될 것이다.

이 논문에서는 프로토타입의 구현에 있어 자료분배관리와 시간관리 서비스는 사용하지 않았으며, 네트워크 환경을 동일한 LAN 상으로 한정하였다.

추후 본 프로토타입 시뮬레이션을 확장하여, 보다 복잡하고 실세계 도메인을 상세히 묘사하는 시뮬레이션을 개발하고자 할 때에는 반드시 자료분배관리 서비스와 시간관리 서비스에 대한 세부적인 연구를 바탕으로 사용하여야 할 것이며, 분산된 상이한 네크워크 환경에서의 실행에 대한 추가적인 연구가 뒤따라야 할 것이다.

참 고 문 헌

[1] Graham Beasley, Distributed Interactive Simulation Whitepaper version 1.0, Silicon Graphics Incorporated Advanced Graphics Division, 1994.

[2] Roy Crosbie, High Level Architecture Advanced Topics, California State University, 1999.

[3] Roy Crosbie, High Level Architecture Basic Concepts, California State University, 2001.

[4] U.S. Department Of Defense, High Level Architecture Federation Execution Details(FED) File Specification, DMSO, 1998.

[5] U.S. Department Of Defense, High-Level Architecture Interface Specification Version 1.3, DMSO, 1998.

[6] U.S. Department Of Defense, High Level Architecture Object Model Development Toos User's Guide, DMSO, 1998.

[7] U.S. Department Of Defense, High-Level Architecture Object Model Template Specification Version 1.3, DMSO, 1998.

[8] U.S. Department Of Defense, High Level Architecture Federation Development and Execution Process Checklists version 1.5, DMSO, 1999.

[9] U.S. Department Of Defense, High Level Architecture Federation Development and Execution Process(FEDEP) Model , DMSO, 1999.

[10] U.S. Department Of Defense, RTI-NG 1.3v3.2 Installation Guide, DMSO, 2000.

[11] U.S .Department Of Defense, RTI1.3-Next Generation Programmer's Guide Version 3.2, DMSO, 2000.

[12] U.S. Department Of Defense, RTI-NG 1.3v3.2 Release Notes, DMSO, 2000.

[13] U.S. Department Of Defense, RTI Development History, DMSO, 2000.

[14] U.S. Department Of Defense, "AMG-41, Chang-Jo 21(Republic of Korea)", DMSO, 2001.

[15] 윤석준, "시뮬레이션 기술의 새 지평 DIS(Distributed Interactive Simulation)", 월간항공, 1995.

[16] 윤석준, "High Level Architecture-시뮬레이션

의 새 지평", 항공산업연구 53, 2000.
[17] 장상철 외, 국방모의분석체계 구축을 위한 상위체계구조(HLA) 기술 연구, 한국국방연구원, 1999.