대한산업공학회/한국경영과학회 2002 춘계공동학술대회
한국과학기술원(KAIST) 2002년 5월 3일-4일

상위체계구조를 사용한 수송모형 연구

이 상헌
국방대학교 운영분석과

A Study for the Transportation Prototype Model Based on the High Level Architecture

Sang-Heon Lee
Dept. of Operations Research, Korea National Defense University, Seoul

The High Level Architecture(HLA) for modeling and simulation was developed as means of facilitating interoperability among simulations and promoting reuse of simulations and their components. The purpose of this paper is to provide a summary of the latest release of the HLA concept, supporting utilities and develop the prototyped Transportation Movement Management(TMM) federation. To obtain this goal, the Federation Development and Execution Process(FEDEP) is being applied to development of TMM federation will consist of three federates. This paper outlines the rationale of our approach, describes the application of the FEDEP in the development of the federation, and provides the current status of the federation development. The resulting federation shows complete interoperability among simulation components in the TMM federation and satisfactory simulation outputs. We present a description and process of the federation and the lessons learned with the process utilization for federation development and execution. Furthermore, the issues in establishing a HLA based federation across multiple legacy simulations are discussed.

1. 서 론

1.1 연구배경

시뮬레이션 분야에 있어서 산업구조가 복잡하고 다양해짐에 따라 특정 시스템을 독립적으로 운영하는 경우보다 하부 시스템들 간에 자료 교환을 통하여 운영하는 경우가 대부분으로 실제 컴퓨터로 완전한 상호운용성(interoperability)을 구현하기가 매우 제한을 받고 있는 실정이다. 이를 극복하기 위한 대표적인 방법이 분산 시뮬레이션(DIS:Distributed Interactive Simulation)이다. 분산 시뮬레이션은 서로 이격된 시뮬레이션들 간에 필수적인 자료교환을 위하여 컴퓨터 네트워크 기술을 개발하여 활용하고 있다. 그러나, 이러한 분산 시뮬레이션들은 응용 기술 및 개발 도구가 상이하면 동종류 시뮬레이션이더라도 상호운용성을 유지할 수 없고 빈번한 개념 및 기술진화에 따라 일정한 시간경과 후 다시 사용할 수 있는 재사용성(reusability)이 결여되어 있기 때문에 많은 개발비용 및 시간을 낭비하는 단점을 갖고 있다.

최근 미 국방성(DoD : Departure of Defense)은 이러한 문제점을 극복하기 위하여 분산된 시뮬레이션들간 개발 언어 및 운영체계가 서로 동일하지 않더라도 연동이 가능한 상호운용성과 시뮬레이션 구성요소 수정에 대처하기 위한

재사용성을 향상시키기 위하여 상위체계구조(HLA:High Level Architecture)라는 차세대 분산 시뮬레이션의 새로운 개념을 제시하였다[1,12].

HLA하 새로운 시뮬레이션 아키텍처는 미 국방성이 중심이 되어 시뮬레이션 적용 범위를 전술 교리 개발, 합동 훈련, 무기체계 획득분야 등 광범위한 분야로 확장시키며 HLA로 전환 노력을 집중하고 있다. 더구나 HLA하 시뮬레이션은 단순히 군사분야 뿐 만 아니라 연방정부, 공공기관, 산업계 및 학문 분야에도 표준화하여 채택될 예정이기 때문에 그 파급 효과의 중요성은 실로 막대하여 국가적인 차원에 이른다.

1.2 기존 연구 고찰

군사 및 관련학계에서는 HLA의 확장성 및 성능의 효과를 인지하였고, 이미 국방 및 학문 분야의 시뮬레이션에서 HLA는 필수적인 항목으로 고려되기 시작하였으며 현재는 물류, 의료, 생산시스템 등의 일반 사회분야의 시뮬레이션에도 HLA를 적용하려는 노력을 쉽게 발견하기에 이르렀다[9,10,13]. 이들 연구의 공통적인 결론은 학계와 산업구조 전반에 미치는 영향이 상당할 것이라고 예측하였다.

본 논문의 목적은 시뮬레이션 아키텍처 기술인 HLA에 근거하여 미 국방성이 적용하고 있는 표준개발과정에 따라 수송이동관리 시스템을 실례로 프로토타입 모델을 개발함으로서, HLA에 대한 개념과 기술 개발에 대한 사례 연구이다.

2. HLA 개요

2.1 HLA 구성요소

HLA는 기존 시뮬레이션 자료의 송수신을 확대함으로써 모든 시뮬레이션들간의 상호운용성을 촉진시키고, 시뮬레이션의 코드와 아키텍처 기본틀(framework)의 재사용성을 증가시키는 것이다. 이를 위해, 각 페더레이트간 상호운용성을 보장하기 위해서 사용하는 공통 구성요소 기반인 RTI와 페더레이트 사이의 상호관계를 명시하여, 페더레이트간의 자료 교환 및 전달 방법을 정의함으로서 자료의 일관성을 유지하는 필수적인 요소인 접속 규약(Interface Specification), 페더레이션 실행간에 각 페더레이트의 상호작용 및 상호운용성을 보장하기 위해 준수해야 하는 객체모델간의 준수사항인 규칙(rules), HLA하 객체 모델 정보를 표현하기 위한 공통방법으로서 페더레이션과 페더레이트 객체모델을 표현하기 위하여 표준화된 표기 형식을 지정하는 객체모델양식(OMT:Object Model Templete)의 구성요소를 가지고 있다[3]. .

2.2 HLA 개발 지원도구

2.2.1 표준개발절차(FEDEP:Federation Development and Execution Process)

HLA에 근거한 시뮬레이션을 개발하기 위한 표준개발과정은 기본적으로 다음과 같이 여섯 단계로 구성되어 있다[3].

- 1 단계:페더레이션 목적 정의(define federation objective)
  사용자와 개발 팀간에 공동으로 상호 달성해야할 페더레이션의 목적과 요구사항을 규정하고 동의한다.
- 2 단계:페더레이션 개념모형 개발(develop federation conceptual model)
  문제공간내 특성에 기초를 실제 도메인에 대하여 적합하게 묘사해야 할 시스템의 개념모형을 개발한다.
- 3 단계:페더레이션 설계(design federation)
  페더레이션내에서 각 임무별, 기능별 페더레이트들 할당하기 위하여 페더레이트들간에 상호 역할을 결정하고 설계한다.
- 4 단계:페더레이션 개발(develop federation)
  페더레이션 객체모델을 개발하고 데이터베이스 및 알고리듬에 페더레이트가 접근할 수 있는 객체모델인 페더레이션 객체모델(FOM), 시뮬레이션 객체모델(SOM)을 개발, 수정 및 보완한다.
- 5 단계:페더레이션 통합 및 시험(integrate and test federation)
  페더레이션을 구성하는 모든 페더레이트에 대해 시험하고 페더레이션 요구량 보장 여

부를 확인한다.

- 6 단계:페더레이션 실행 및 결과 분석 (execute federation and prepare result)
  페더레이션을 실행하고 산출 결과에 관하여 목적 및 요구사항 달성 여부를 분석한다.

본 논문에서 제시한 수송이동관리 시제 모형은 상기의 표준개발절차(FEDEP)에 의해 개발하였다.

2.2.2 RTI(RunTime Infrastructure)

RTI는 HLA 접속 규약에 의해 구체화된 서비스를 페더레이션에 참가한 모든 페더레이트에게 제공하는 소프트웨어이다. RTI는 페더레이션을 생성하고, 페더레이트가 페더레이션에 참여하고 탈퇴하는 것을 관리하여, 참여중인 페더레이트들간에 자료 교환을 용이하게 하는 FedExec (federation executive process), FedExec 생성과 소멸을 관리하여 페더레이트에게 적합한 페더레이션 실행에 참여하는 것을 지시하는 RtiExec(RTI executive process), 접속 규약에 명시된 서비스를 페더레이트 개발자에게 제공하는 libRTI(library RTI)로 구성되며, 페더레이트들은 libRTI내 RTIambassador 구조로 RTI에 의해 제공되는 서비스와 그 서비스에 대한 응답 기능을 가진 Federateambassador로 이루어진 LRC(local RTI component)를 통하여 RTI와 페더레이트들간에 서로 통신한다[5,7]. 본 논문에서는 RTI-1.3NG version 3.2를 사용하였다.

2.2.3 OMDT(Object Model Development Tool)

OMDT는 페더레이션 개발 과정에서 중요한 요소인 페더레이트의 시뮬레이션 객체 모델(SOM)과 페더레이션의 페더레이션 객체 모델(FOM)을 자동적으로 생성하고 HLA 객체모델양식에 정의된 객체모델과 일관성을 유지시켜주는 소프트웨어이다.

OMDT는 HLA OML(Object Model Library)을 이용하여 관심 있는 객체모델에 대하여 참고할 수 있고 일단 객체모델을 구축한 후에 페더레이션을 생성하기 위해 RTI에서 요구되는 fed파일을 제공한다[4]. 본 논문에서는 OMDT version 1.3을 사용하였다.

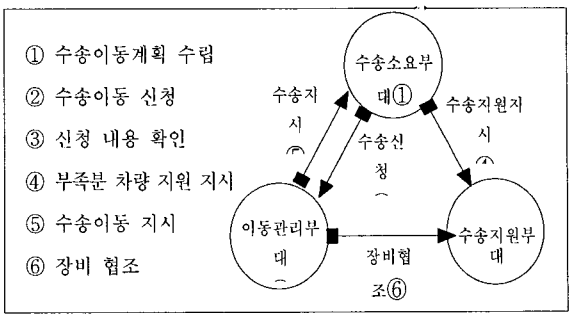
3. 수송이동관리 개념 모형

3.1 수송이동관리 페더레이션 요구사항

수송이동관리란 병참선을 이동하는 모든 병력, 장비, 보급품 등의 이동소요를 최대한 충족시킬 수 있도록 수송능력을 경제적이고 효과적으로 사용하기 위하여 수송능력과 이동 노정을 계획, 지시하고 수송능력 운영을 협조, 통제 및 조정하는 활동을 의미한다.

[그림 1]은 본 수송이동관리 페더레이션(TMM :Transportation Movement Model)의 개략적인 요구사항을 보여주고 있다. 수송이동관리 모형에 표현해야 할 객체는 수송이동관리에 관련된 부대들이다. 수송소요부대가 필요한 소요를 결정하는 1 단계, 수송능력을 판단하는 2 단계, 이동 우선 순위를 결정하는 3 단계, 수송이동 계획을 작성하는 4 단계를 거쳐 수송이동계획을 작성 후 이동관리부대에 수송이동 신청을 한다. 이동관리부대는 수송이동의 시점부터 종점에 걸쳐 부대의 이동, 도로 등의 모든 수송이동 제반 정보를 상시 유지하여 그 신청 내용을 시간별, 노정별 등으로 검토, 조정하여 신청부대에 수송 이동을 지시한다. 만약, 수송소요부대의 차량대수가 부족하면 수송지원부대에 차량지원을 지시하고, 지시 받은 수송지원부대는 수송소요부대와 협조하여 기 수립된 수송계획에 의거 이동한다.

본 수송이동관리 시제 모형은 HLA에 근거하



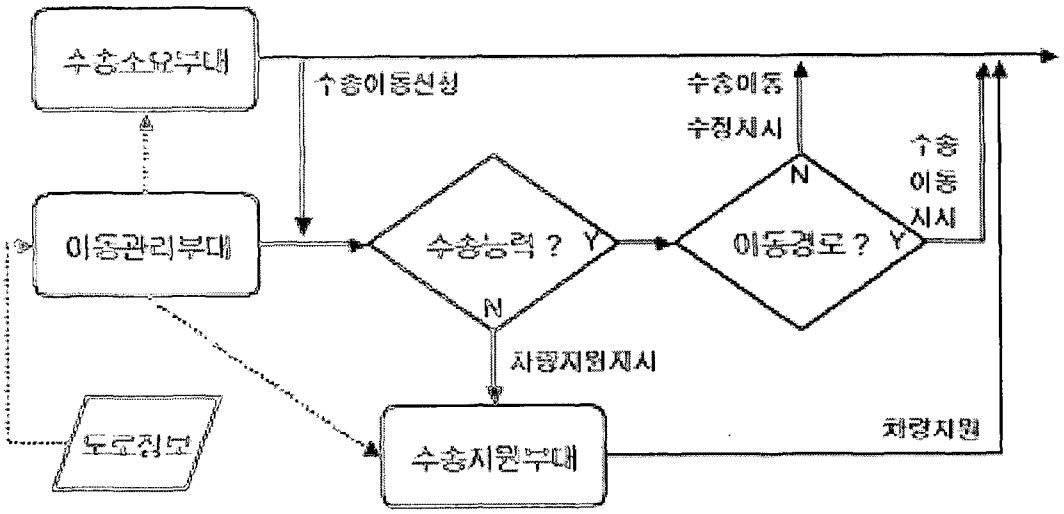
[그림 1] 수송이동관리 페더레이션 요구사항

여 표준개발절차에 의한 페더레이션 개발 등을 최우선 목적으로 하기 때문에, 수송이동관리

시스템의 복잡성을 가능한 최소화하여 HLA하에서 단순한 수송이동관리 모델로 제한한다.

3.2 수송이동관리 개념모형

[그림 2]는 수송이동관리 시스템 절차를 나타내는 개념모형이다. 클래스로 묘사된 부대들과 이 부대들의 활동공간인 도로망을 기초로 수송이동관리 모델의 원활한 상호작용을 하기 위해 페더레이션 설계 이전에 시스템 내에서의 객체와 상호작용의 흐름을 정의한다.



[그림 2] 수송이동관리 개념모형

본 수송이동관리 페더레이션에서 이동관리부대, 수송소요부대 및 수송지원부대들이 서로 상호작용으로 묘사하기 위하여 부대 속성들을 규정한다. 이동관리부대는 부대명과 부대위치의 속성을 갖고 상호작용으로는 수송지시, 차량지원지시가 있으며, 수송소요부대는 부대명, 부대위치와 차량보유대수의 속성 및 상호작용으로는 수송이동신청이 있으며, 수송지원부대는 부대명, 부대위치, 차량보유대수, 출동가능대수의 속성 및 수송신청, 차량지원협조의 상호작용으로 규정한다.

4. 수송이동관리 페더레이션 설계 및 개발

4.1 수송이동관리 페더레이션 실행 환경

[그림 3] 수송이동관리 페더레이션 실행 환경

수송이동관리 페더레이션은 [그림 3]과 같이 동일한 네트워크 상의 분산 환경 하에서 하나의 페더레이션으로 구축하여 RtiExec를 통한 객체간의 상호작용으로 수송이동관리를 표현하고자 한다. HLA 연동을 위한 가정사항들은 다음과 같다.

첫째, 모델의 적용을 위한 환경은 윈도우 NT 4.0과 윈도우 98 O/S PC로 한정한다.

둘째, 생성하는 객체는 하나로 제한한다.

셋째, 수송지원부대는 중대 급으로 한정한다.

넷째, 도로 속성은 부대 클래스에 포함하여 표현되도록 한다.

4.2 RTI 함수에 의한 페더레이션 알고리즘

수송이동관리 페더레이션은 시나리오 전개 및 수명주기 관점에서 크게 준비(startup), 운영(operation), 종료(shutdown)의 세 가지로 나타낼 수 있다. 준비 단계에서는 수송이동관리 페더레이션의 생성과 각 부대들의 객체 선언,

객체의 생성 및 등록, 객체 발견과 속성 초기화 등이 있으며, 운영 단계에서는 객체 속성과 상호작용 반영, 수송이동관리 페더레이션 시간관리, 객체의 소유권 교환, 객체의 제거 및 생성 등이 있으며, 종료 단계에서는 수송이동관리 페더레이션에서 탈퇴와 종료 등이 있다. 이와 관련된 RTI 함수들을 선정하여 페더레이션에 대한 초기에 알고리듬을 구성한다[13].

수명주기에 의하여 수송이동관리 페더레이션과 사용한 RTI 함수를 선정하여 페더레이션의 알고리듬을 구성하면 다음과 같다.

- 1 단계:수송이동관리 페더레이션 실행
- 2 단계:페더레이트 자료 초기화
  - RTIambassador 객체 생성
  - Federateambassador 객체 생성
  - 실행시간간 요구되는 특정객체 초기화
- 3 단계:페더레이션 생성 여부
  - createFederationExecution() 호출
  - 페더레이트 암바세도어 등록
  - FederationExecutionAlreadyExist()를 RTI 암바세도우에서 호출하면 5단계로 진행
- 4 단계:페더레이션 생성
- 5 단계:페더레이션 참가
  - joinFederationExecution() 호출
  - 에러 확인-결정적인 에러 발생시 종료
- 6 단계:ID값 획득(publish and subscribe)
  - getObjectClassHandle() 호출
  - getAttributeHandle() 호출
  - RTI::AttributeHandleSetFactory::create() 호출
  - subscribeObjectClassAttributes() 호출
  - publishObjectClass() 호출
- 7 단계:객체 속성 최신화
  - updateAttributeValues() 호출
  - theAttributes.getValue()로 최신화
- 8 단계:RTI에서 사건이 발생하는가?
  - tick() 호출
  - 사건이 존재하면 9단계로 진행
  - 사건이 존재하지 않으면 10단계로 진행
- 9 단계:tick()하고, 8단계로 진행
- 10 단계:tick() 동안 페더레이트 사건을 종합
- 11 단계:종료할 것인가? 아니면 7단계로 진행
- 12 단계:페더레이션 종료.
  - resignFederationExecution() 호출
  - destroyFederationExecution() 호출

4.3 수송이동관리 페더레이션 개발

본 수송이동관리 페더레이션의 주 클래스는 부대 클래스이고, 하부 클래스로 수송지원을 위한 차량지원 클래스, 이동지시/접수하는 수송이동신청 클래스, 이동노정을 위한 노드와 호를 표현과 수송이동을 묘사하기 위한 좌표는 피지원부대 클래스, 도로정보의 교환을 위한 우발상황 클래스 등이 있다.

5. 수송이동관리 페더레이션 통합 및 실행

5.1 페더레이션 통합 및 시험

5.1.1 페더레이션 관리

실제적인 시뮬레이션을 수행하기 전에 페더레이션 공통 환경을 구축하고, 기본 객체 생성을 위한 RtiExec를 실행한다. 페더레이션을 생성하기 위하여 createFederationExecution()을 이용하여 객체모델양식에서 이미 설정된 페더레이션 실행 이름을 기준으로 페더레이션에 적합한 페더레이션 실행환경을 구축한다. 이때, RtiExec가 fedexec로 변화한다는 것은 페더레이션내에 공통 환경을 구축함을 의미한다. 페더레이션 내에 RtiExec는 유일하지만 RtiExec 내에서 발생할 수 있는 fedexec는 여러 개가 될 수 있다. 입력 변수는 페더레이션의 이름과 공통 환경을 위한 fed 파일이다. 공통 환경이 구축되면 joinFederationExecution()이 수행된다.

joinFederationExecution()은 페더레이션 실행명과 자신의 페더레이트 실행명을 사용하여 RTI내에서 페더레이트 자신의 ID를 할당받게 된다.

resignFederationExecution()은 페더레이트가 페더레이션에서 탈퇴할 때, 페더레이션에서 마지막으로 탈퇴하는 페더레이트가 페더레이션을 종료할 때 destroyFederationExecution()을 호출한다. 따라서, joinFederationExecution(), resign

FederationExecution()의 사이에 페더레이션 실행이 될 수 있도록 유도해야 한다[2,4,5,6].

5.1.2 자료전송을 위한 객체속성 설정

페더레이션에 참가한 특정 페더레이트가 페더레이션 실행 동안 상호작용을 원활히 하기 위하여 객체를 선언하는 과정이다. 객체의 속성을 publish와 subscribe 옵션을 설정하여 수행하는 과정을 표시한다. PublishObjectClass()를 사용하여 클래스를 publish 할 수 있고, subscribe 할 수 있다면 페더레이트는 그 클래스내의 정보를 이용하여 반응할 수 있다. publish에서는 PublishObjectClass()를 사용하여 자신이 다른 페더레이트에게 전송할 객체의 종류 및 속성들의 크기를 가지고 페더레이션 실행에 등록한다. 이러한 과정을 걸쳐 등록 된 후 다른 페더레이트에서 자료를 참조할 수 있게 된다. 이와 덧붙여 subscribe에서 자신이 수신하게 될 객체의 종류 및 속성의 크기를 가지고 Subscribe ObjectClassAttributes()를 사용하여 마찬가지로 페더레이션 실행에 등록한다.

특히 subscribe가 완료된 후 다른 페더레이트에서 publish하면 자동적으로 subscribe의 등록여부에 따라 페더레이트간에 자료 교환이 성립된다. 이때 RTI의 함수 DiscoverObject Instance()가 작용한다. 이 기능에서는 다른 페더레이션이 등록했을 때, 사용한 객체의 이름, 객체 클래스 ID 및 객체의 인스턴스 ID를 수신한다.

이 과정들을 수행하여야 페더레이트의 상호작용을 위한 기반구조 설정이 완료된다. 이후에 페더레이션에서 발생한 자료들은 송신(publish)하고 수신(subscribe)하는 기능이 수행된다[2,4,5,6].

5.1.3 자료 전송

시뮬레이션 수행 중 다른 페더레이트로 공포(announce)해야 할 객체의 속성을 최신화하기 위해 UpdateAttributeValues()의 기능을 이용한다. 이 기능은 AttributeHandleValue Pairset()과 자신이 가지고 있는 객체 ID를 이용하여 다른 페더레이트로 정보를 전달하는 임무를 수행한다.

AttributeHandleValuePairset()은 그 페더레이트 자신이 전송할 세부 속성값들을 가지고, 메시지 전송은 ParameterHandleValuePairset()을 사용한다. 물론 다른 페더레이트로 메시지나 객체 속성값을 전송하기 위해서는 publish 기능이 선행되고, 수신 페더레이트에서는 subscribe 기능이 선행되어야 한다. 다른 페더레이트에서 자료 전송이 있으면 ReflectAttributeValues()의 기능을 수행하여 전송 받은 자료를 분류한다. 수신된 자료에는 발신한 페더레이트의 정보와 그곳에서 보낸 각 객체 속성 정보를 포함하고 있다[2,4,5,6].

5.1.4 시간 동기화

자료의 송수신과 더불어 RTI 및 분산 시뮬레이션에서 가장 중요한 항목 중의 하나가 시간의 동기화(synchronization)이다. 이러한 시간 동기화 문제를 RTI에서는 시간관리 항목에서 처리한다. 특히 강조되는 항목으로는 규제(regulating) 및 구속(constrained)이다.

규제(regulating) 시간으로 정의된 페더레이트는 어느 특정 시간에 사건을 발생하는 TSO(Time-Stamp-Order) 사건(event)을 발생할 수 있는 능력을 보유하고 있다. 구속(constrained) 시간으로 정의된 페더레이트는 TSO 사건을 수신할 수 있다. 시뮬레이션 모델의 특성에 따라서 두 가지 항목이 적절히 선언되어야 원활하고 적절한 시간 관리가 될 수 있다.

본 논문에서 시간 진행방법은 일정시간 진행방법과 사건기반시간 진행방법의 혼합형을 사용하였다. 일정시간 진행방법은 timeAdvance Request()의 기능을 사용하여 RTI에서 시간진행을 요구한다. timeAdvanceRequest()를 전송하면 timeAdvanceGrant()를 수신하여 다음 시간으로 진행한다. 반면에 사건기반 시간 진행방법을 사용하는 페더레이트에서는 nextEventRequest()의 기능을 사용하여 마찬가지로 timeAdvance Grant()를 수신한다[2,4,5,6].

5.2. 페더레이트 시험

[그림 4]는 페더레이트가 페더레이션 생성, 참가 및 객체 속성 등을 시험한 결과이다. 표식 "1"은 노드 2에서 노드 10까지의 최단 이동경로를 보여주고 있다. 노드 2에서 [0,2], 노드 7에서 [40,2], 노드 10에서 [90,7]의 형태로 되어 있다.

[a,b]에서 a는 해당 노드까지의 최단 이동시간을 나타내며, b는 최단 이동시간이 되게 하는 이전 단계의 노드이다. 노드 2에서 노드 10까지의 최단 이동시간은 90단위시간이고 이동경로는 표식 "2"에서와 같이 2-7-10이다. 이 이동경로는 이동관리부대에서 수송신청부대에 수송이동지시를 하게된다. 그리고 실행간의 도로의 속성값의 변화에 따라 실시간으로 최단경로의 소요시간을 산출할 수 있다.

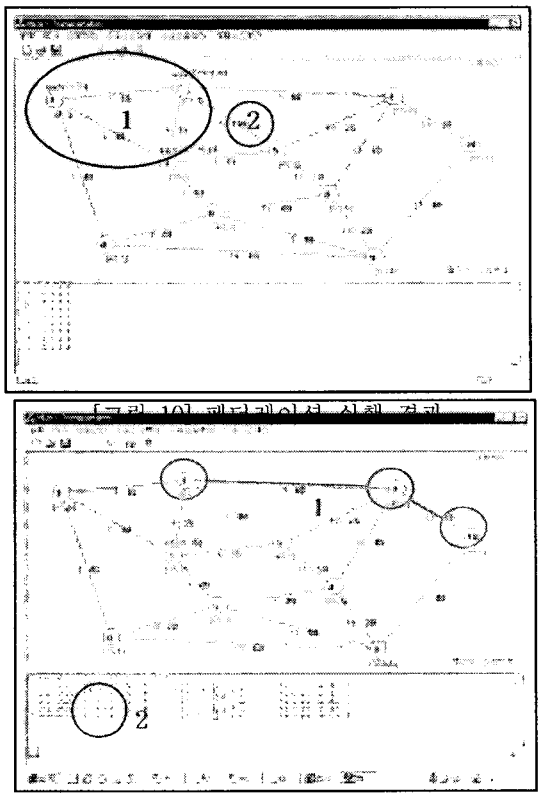
5.3 페더레이션 실행 및 분석

[그림 5]는 이동관리부대 페더레이트 화면의 페더레이션 실행 결과를 나타낸다. 표식 "1"을 통하여 각 부대 페더레이트가 부대 객체들의 속성들이 서로 RTI를 경유하여 페더레이션에 참가됨을 확인할 수 있고, 표식 "2"는 이동관리부대 페더레이트가 우발상황 메뉴를 통해 도로 정보가 변화되었음을 보여준다. 이 도로정보 속성의 변화로 1번 노드에서 10번 노드까지 이동시간 계산 결과는 [그림 5]에서 확인되는 것처럼 95이다. 이는 다익스트라 알고리즘[11]을 이용하여 산출된 결과와 일치하는 현상을 보여주고 있다.

이를 RTI 함수관계로 살펴보면, 이동관리 페더레이트에서 페더레이션 이름을 'Transportation', 각 객체의 속성들을 정의한 'Troop.fed'를 사용하여 createFederationExecution() 함수로 인해 페더레이션 생성, 페더레이트 이름을 각각 부대 명으로 하며, 페더레이션 이름을 'Transportation'으로 하며, RTI의 콜 함수인 페더레이트 암바세도우에 대한 포인트 지정으로 joinFederationExecution()으로 페더레이션 참가가 이루어졌다. 자료 전송은 updateAttributeValues() 함수로 각 페더레이트는 getValue()로 속성을 최신화한다.

enableTimeConstrained()와 enableTimeRegulation() 함수로 시간관리는 원활히 동기화로 일정시간 5분에 대해 변화하는 것을 [그림5]의 에디터 부분에서 확인 할 수 있다. 본 수송이동관리 페더레이션에서는 NextEventRequest() 함수로 사건 진행 원활히 됨을 알 수 있다. 각 사건들은 객체들의 생성 및 페더레이션의 참가, 수송이동지시와 도로속성값을 변화하는 우발상황 등 사건들이 있다. 특히 이동시간계산은 이동관리 페더레이트에서 우발상황 속성값의 변화에 따라 실 시간적으로 변화함을 다른 페더레이트에서 확인할 수 있다. 이런 사건들의 실시간 요소들의 시간적 통제에서 에러들이 발견된 바 있다. 시간통제면에서 발생한 문제들은 표준개발절차에 따라 환류하여 3단계(design federation)에서 페더레이트의 상호작용 클래스 할당에 대한 점검, 4단계에서 FOM, SOM 및 알고리듬 점검, 5단계에서 RTI 함수, 프로그램 코딩에 대한 확인 점검하는 과정을 반복적으로 수행해야 한다.

본 수송이동관리 페더레이션을 실행한 결과



[그림 4] 이동관리 페더레이트 시험 결과

HLA 기반인 RTI하 객체들의 속성들이 서로 RTI를 통해 성공적인 자료 교환을 실시하여 수송이동 신청, 차량지원 지시 등 상호작용들의 파라메터를 연결시키는 상태가 만족한 것으로 분석된다.

6. 결 론

본 논문에서는 미 국방성에서 진행하고 있는 국방 모델링 및 시뮬레이션 분야에서 가장 핵심 과제 중에 하나인 HLA에 대한 개념, 지원 서비스, 그리고 표준개발절차인 FEDEP에 대해 고찰하였으며, HLA 기반 하에 FEDEP에 따라 시제형 수송이동관리 페더레이션을 구현하였다.

본 연구를 통하여 HLA를 기반으로 시뮬레이션을 개발할 경우 상호작용에 필요한 기본 하부구조가 RTI를 통해 자동적으로 연동되어 개발이 용이하고, 또한 동일한 하부구조와 시뮬레이션 구성요소들을 사용함으로써 페더레이션 실행간 상호운용성이 보장될 수 있으며, 또한 모델 개발시 구성요소 단위로 페더레이트가 설계되어 기 개발된 객체 또는 구성요소들을 재사용하기가 용이함을 알 수 있었다. 이와 같은 다수의 장점들을 효율적으로 활용하면 대규모 모델링 및 시뮬레이션 개발시 막대한 개발비용을 절감할 수 있고, 이와 병행하여 개발기간을 크게 단축시킬 수 있는 간접적인 경험을 체험할 수 있었다. 특히 실제 시스템과 사람이 참여하는 분산 가상 환경 속에서 벌어지는 각종 시스템의 통합에서 핵심적인 역할을 주도할 것으로 예상된다.

7. 참고문헌

[1] 이상헌, "HLA 모의구조 전환에 따른 한국군 DM&S 발전방향", 「한국국방경영분석학회지」, 제26권, 제2호(2000).

[2] Department of Defense, *High Level Architecture FederateAmbassador Reference RTI 1.3 Version 1.3*, USA, 1998.

[3] Department of Defense, *High Level Architecture Federation Development and Execution Process (FEDEP) Model Version 1.4*, USA, 1998.

[4] Department of Defense, *High Level Architecture Object Model Development Tool(OMDT) User's Guide Version 1.3*, USA, 1998.

[5] Department of Defense, *High Level Architecture RTIAmbassador Reference RTI 1.3*, USA, 1998.

[6] Department of Defense, *High Level Architecture Run-Time Infrastructure Programmer's Guide Version 1.3*, USA, 1998.

[7] Department of Defense, *High Level Architecture Supporting Class Reference RTI 1.3*, USA, 1998.

[8] Frederick Kuhl, Richard Weatherly, and Judith Dahmann, *Creating Simulation Systems : An Introduction to the High Level Architecture*, Prentice Hall, USA, 2000.

[9] Mikel D. Petty, Pitor S. Windyga, "A High Level Architecture-Based Medical Simulation System", *Simulation,* Vol.73, No.5(1999), pp 281-287

[10] Lightner. M. and D. Judith, "The High Level Architecture for Simulation", *Simulation* Vol.73, No.4, (1999)

[11] Phillips. D. T, and A. Garcia-Diaz, *Fundamentals of Network Analysis*, Englewood Cliffs, Prentice-Hall, 1981.

[12] Steven D. Knight, "A Comparison of Analysis in DIS and HLA", Naval Postgraduate School, Monterey USA, 2000.

[13] Thomas Schulze, Steffen Stanβburger, Ulrich Klein, "Migration of HLA into Civil Domains : Solution and Prototypes for Transportation Applications", *Simulation,* Vol.73, No.4(1999), pp 296-303