IE Interfaces
Vol. 17, No. 1, pp. 128-141, March 2004.

상위체계구조를 이용한 컨테이너 터미널 운영방안 연구

이상헌† · 이찬우

국방대학교 운영분석학과

A Study on the Operational Plan for Port Container Terminal Using High Level Architecture

Sang Heon Lee · Chan Woo Lee

Department of Operations Research, National Defense University, Seoul, 122-875

Although a number of container terminal simulators have been developed for various purposes, none of the existing simulators allow the system structure to reuse the system structure depending on application. Our goal is to develop highly reusable, highly inter-operable and flexible container terminal simulation system. The High Level Architecture(HLA) is an architecture for reuse and inter-operation of simulation. It is based on premise that no simulation can satisfy all use and users. An individual simulation or set of simulations developed for one purpose can be applied to another application under the HLA concept of the federation : a composable set of interacting simulations. The intent of the HLA is a structure which will support reuse of capabilities available in different simulations, ultimately reducing the cost and time required to create a synthetic environment for a new purpose, and the possibility of distributed collaborative development of complex simulation applications. In this paper, we discuss the design of a HLA-based port container terminal simulation system. Furthermore, we describe various technical motivations for HLA, the key elements of the architecture and how they are minimum and essential to the goal of reuse and interoperability.

Keywords: high level architecture, HLA, container terminal, seaport operation, simulation, reusability, interoperability

1. 서 론

1.1 연구배경 및 목적

우리나라는 지속적인 경제성장과 더불어 항만물동량이 연평균 수입 11.9 %, 수출 14.1 %, 그리고 연안화물은 14.7 %씩 꾸준한 증가추세를 보이고 있다. 뿐만 아니라 화물수송에 있어서 컨테이너화가 급속하게 진전되어 수출·입 물동량에 있어서 컨테이너에 의한 화물수송의 비중이 크게 증가하여 왔고, 이에 따라 컨테이너 터미널 능력에 대한 연구 또한 지속되어 왔다.

항만하역 능력 산출방식에는 확정적 접근방법과 확률적 접근방법으로 나누어 볼 수 있으며, 확률적 방법에는 대기이론(queueing theory)과 시뮬레이션 기법이 있다. 확정적 접근방법은 항만하역 능력에 영향을 주는 매개변수 값들을 확정적인 값으로 간주하고 개략적으로 하역능력을 산출하는 방식으로 우리나라 주요 항만의 공칭 하역능력 산정에 사용되고 있다.

대기이론은 주로 서버(server)가 단일 종류이고 도착분포와 서비스분포가 주어진 경우에 대기시간이나 자원들의 점유율 등을 추정하여 사용된다. 다수의 항만개발 투자보고서에서는 선박의 도착분포와 서비스 시간이 지수분포임을 가정하여 사용하고 있으나 실제 항만 시스템은 부두, 하역장비 등 다단계로 이루어진 복잡한 시스템이기 때문에 대기이론은 현실적인

†연락저자 : 이상헌 교수, 122-875 서울시 은평구 수색동 205번지 국방대학교, Fax : 02-309-9774, E-mail : leesangh@kndu.ac.kr
2003년 1월 접수, 1회 수정 후 2004년 2월 게재 확정.

결과를 제공하지 못한다(김창곤 외, 2000).

시뮬레이션 기법은 주로 동적이고 복잡한 실제 시스템을 컴퓨터 모델링함으로써 이를 통해 시스템 특성을 파악하고 시스템 성능을 평가함으로써 시스템의 설계, 운영 및 개선에 기여할 수 있는 유용한 기법이라고 정의된다(장성용 외, 1988). 시뮬레이션 기법은 항만의 동적이고 복잡한 시스템을 현실적으로 모델링함으로써 보다 정확한 능력의 산출이 가능하다. 항만에서의 주요 응용분야는 항만개발 분야와 항만운영 분야로 나누어 볼 수 있다. 항만개발 응용에서는 단위부두 혹은 터미널의 개괄적인 시뮬레이션과 항만 전체를 종합적으로 고려한 시뮬레이션 모형이 있다. 항만운영 분야의 응용은 기존의 항만 혹은 부두의 운영효율을 높이기 위해 장비확충, 하역시스템 개선 등이 주된 목적이다. 컨테이너 터미널의 경우 건설계획 수립, 최적 운영 시스템 개선 등을 위한 시뮬레이션 모형 등이 개발되어 사용되고 있는 실정이다.

지금까지의 연구현황을 살펴보면 장성용, 박진우(1988)는 SIMAN을 적용하여 컨테이너 터미널의 적정능력을 산출하는 방법을 연구하고 부산항의 기존 컨테이너 터미널과 계획이 확정되어 추진중인 컨테이너 시설들의 적정능력을 예측하여 향후 컨테이너 물동량에 대한 대처방안을 제시하였으며, 한국해양수산개발원(김창곤 외, 2000)은 컨테이너 터미널 안벽에서의 작업능력에 영향을 미칠 수 있는 다양한 변수들의 확률적인 요인 및 운영현황을 분석하고, 이러한 분석에 근거하여 컨테이너 터미널 안벽에서 이루어지는 상황을 객체지향 프로그램 개발환경을 제공하는 독일 Gensym사의 시뮬레이션 전용 소프트웨어인 G2(Gensym 2)를 이용하여 적정 안벽능력을 분석하였다.

그러나 이러한 시뮬레이터들은 복잡하고 다양해진 최근 산업구조가 특정 시스템을 독립적으로 운영하는 경우보다 하부시스템들 간에 자료교환을 통하여 운영하는 경우가 대부분이어서 실제 컴퓨터로 완전한 상호운용성(interoperability)을 구현하기가 매우 제한을 받고 있는 실정이다. 이를 극복하기 위한 대표적인 방법이 분산 시뮬레이션(DIS; Distributed Interactive Simulation)이다(이상헌, 2000).

특히 상위체계구조(High Level Architecture, 이하 HLA)를 기반으로 하는 실시간 분산 시뮬레이션 기술은 합성환경 구축을 위한 핵심기술이다. HLA는 모든 시뮬레이션 상호간 상호운용성을 보장하고 동시에 모델 및 시뮬레이션 구성요소의 재사용성을 촉진시키기 위한 상위수준의 시뮬레이션 아키텍처이다. 즉, 복잡하고 다양해진 산업구조를 시뮬레이션으로 표현하기 위해 필요한 재사용성과 상호운용성을 보장할 수 있다는 것이 HLA를 사용하는 원천적 목적이라고 할 수 있다.

미 국방성을 중심으로 국방 관련 모든 차세대 시뮬레이션 개발에 HLA 표준을 준수하도록 규정하고 있고, 국제전기전자공학회(IEEE)가 시뮬레이션을 위한 표준으로 채택(IEEE 1516)하였으며 그 연구가 민간산업 분야로 빠르게 확산되고 있다. 따라서 HLA가 컴퓨터를 이용한 상호작용 분야에 주도적인 역할을 할 것으로 예상된다. 이에 반해, 국내에서는 아직까지 HLA 기반 실시간 분산 시뮬레이션에 대한 기술 및 연구실적은 매우 미진한 실정이다.

본 연구의 목적은 HLA 기반하에서 연동체계(Run Time Infrastructure, 이하 RTI) 1.3NG-V3.2를 미들웨어로 하고 Visual C++를 이용하여 컨테이너 터미널 시뮬레이터를 개발하고, 적정 컨테이너 크레인(Container Crane, 이하 C/C) 수 및 선석 수를 판단하는 HLA 기반 프로그램의 세부 구현방안에 대한 연구를 통하여 미 국방성에서 표준규약으로 제시한 HLA/RTI 하에서 시뮬레이션의 재사용성과 상호운용성의 우수성을 확인하고 민간산업 분야에서의 HLA 적용 가능성을 모색해 보고자 한다.

1.2 연구범위 및 방법

일반산업 분야에 대한 HLA 시뮬레이션 기술적용과 세부 구현방안 연구를 위하여 분산 환경하에서 운용될 수 있는 체계로써 컨테이너 터미널 체계를 모델링하여 그 프로토타입을 개발하는 것을 연구대상으로 설정하였다.

이러한 프로토타입을 개발하기 위하여 미 국방성에서 제공하는 FEDEP(Federation Development and Execution Process) 모델을 사용한다. 이 모델은 HLA 페더레이션(federation)의 개발과 실행을 위하여 상위수준의 프레임워크를 기술하고 있으며 개발자가 애플리케이션(application)의 요구에 쉽게 접근할 수 있도록 페더레이션의 개발과 실행을 위한 지침을 상술하고 있다.

프로토타입을 개발함에 있어서 FEDEP 모델에 따른 절차를 준용하면서 모델링은 단계별로 객체지향 모델링 언어인 UML (Unified Modeling Language)을 사용해서 수행하며, 프로토타입 시뮬레이션 개발을 위한 언어는 마이크로소프트사에서 제작한 프로그램 제작 툴인 Visual C++ 6.0을 이용하여 C++ 언어로 구현하였다. 개발된 프로토타입에 대해서 HLA 기반으로 구현되어 실행하는 프로그램이 LAN 상에서 적합하게 구동하는지의 여부를 확인한다.

이하 HLA 배경과 목적을 달성하도록 구성된 규약과 기반 기술에 대하여 고찰한 후 컨테이너 터미널 페더레이션의 요구를 식별하고 그 목표를 설정하여 페더레이션을 설계한다. 이러한 설계를 바탕으로 페더레이션 객체 모델(Federation Object Model, 이하 FOM)을 개발하고 HLA 프로그램 구현을 지원하는 미들웨어인 RTI에서 제공하는 다양한 서비스를 분석하여 페더레이션을 개발한 후 페더레이션 실행결과에 대하여 분석함으로써 결론 및 발전방향을 제시한다.

2. 기반기술 고찰

2.1 HLA 개요

HLA는 소프트웨어 아키텍처를 정의한다. 근본적으로 각각의 구성요소들이 독립적인 시뮬레이션으로 된 구성요소 기반

의 시뮬레이션을 지원하는 아키텍처이다(Frederick *et al*., 2000).

이러한 상위수준 아키텍처의 주요 목적은 기존 시뮬레이션들의 재사용성(reusability)과 다른 시뮬레이션들과의 상호운용성의 확보이다. 즉, 기존의 시뮬레이션들을 HLA 규약에 맞도록 구조를 편성하여 HLA 기본 틀(framework) 속으로 가져옴으로써, 과거에 다양한 목적으로 개발되었던 시뮬레이션을 통합하여 객체 구성요소의 시뮬레이션이 아닌 페더레이트(federate) 구성요소 기반의 페더레이션을 개발하는 것이다. 이러한 구성요소로서의 페더레이트란, 각 구성요소 시뮬레이션들을 의미하며, 각 페더레이트는 원래 설계된 페더레이션을 벗어나 다른 페더레이션에서도 사용될 수 있는 시뮬레이션이나 도구의 개념이다.

페더레이션은 페더레이트들과 RTI로 구성된다. 페더레이트는 이러한 구성단위이며 또한 소프트웨어의 재사용 단위이다. 또한 HLA는 네트워크 상에서 분산 시뮬레이션을 효율적으로 관장하고 진행시키기 위한 데이터 추상화 구조를 가지고 있으며, 기반구조로서 RTI 소프트웨어를 지원하고 있다. RTI는 기존의 네트워크 프로토콜인 TCP/IP에서 한 단계 향상된 개념의 기반구조(infrastructure)로서 네트워크 상에서 공유되는 원시 데이터를 과거 단순한 일방적 분배체계가 아닌 데이터 분배, 메시지 인보케이션(invocation)에 의한 해당함수 수행, HLA 규칙에 규정된 임무수행까지 수행하는 양방향 서비스를 제공한다. 추상화와 객체지향 구조에 기반을 둔 양식에서 데이터 관리와 그들의 통합된 기본적인 운용은 추상적인 자료양식이나 객체로 캡슐화된다.

HLA는 기본적인 아키텍처로서 페더레이트와 RTI로 형성되어 있다. 이 구조는 페더레이트에서 RTI로, RTI에서 페더레이트로의 양방향으로 데이터 관리가 요구된다. RTI는 페더레이트 내에 라이브러리(library)의 형태로 RTI 인터페이스 서비스를 제공하고, 또 다른 형태로서 Ambassador라는 클래스 구조로 인터페이스를 제공한다.

또한 HLA는 사건기반(event-driven)의 아키텍처이다. 전통적인 분산 시뮬레이션 아키텍처에서는 분산실행에 참여하는 각 시뮬레이션이 상호간에 PDU (Protocol Data Unit) 형태의 데이터를 송수신함으로써 시뮬레이션 상호작용을 수행한다. 그러나 이와는 대조적으로 HLA는 <그림 1>에서 볼 수 있듯이 각 시뮬레이션이 RTI라는 일종의 공통 소프트웨어 모듈을 통하여 정보를 주고받게 하고, RTI 하에서 모든 참여 시뮬레이션이 통합될 수 있도록 하고 있다(DMSO, 1998).

HLA는 이와 같이 사건기반의 아키텍처로서 마치 윈도우 시스템처럼, 발생되는 사건에 따른 해당 큐(queue)의 선택적인 전달 및 해당 기능의 인보케이션 및 실행을 수행함으로써 여러 함수들의 절차들로써 이루어진 메커니즘을 불러오지 않고 구성요소가 하나 또는 그 이상의 사건에 대해 직접적으로 알 수 있게 한다. 한편, 체계의 다른 구성요소들은 자신에 절차를 관련시켜 사건에 대한 관심을 표명한다. 사건이 발생될 경우 체계는 사건을 위해 계획했던 모든 절차를 시행한다. 이러한 메시지 인보케이션은 RTI의 서비스 설계를 광범위하게 적용할 수 있게 한다.

모든 페더레이트는 RTI를 통해서 상호 작용할 수 있다. 페더레이트들이 RTI를 통하여 자신이 필요한 데이터에 대하여 요구(subscribe)하고, 자신이 다른 페더레이트에 줄 수 있는 데이터를 공표(publish)함으로써, 각각의 페더레이트가 자체 프로세스를 수행하면서 RTI 상의 서비스를 실시하게 한다. 그러므로 HLA 페더레이트의 소프트웨어적 구성은 페더레이트가 RTI 기능을 수행하게 하는 RTI 구현 부분과 순수 자체 프로세스를 위한 부분, 그리고 RTI로부터 서비스를 받을 수 있도록 해 주는 인터페이스 부분으로 나눌 수 있다. 이러한 세 가지 부분은 HLA 규칙으로써 묶여 있으며 구조화되어 있다.

상호운용성은 이러한 규칙 속에서 세 가지 부분이 올바르게 정립되어야 보장될 수 있으며, 이러한 구현 패턴은 많은 다른 시뮬레이션 응용 프로그램에 사용될 수 있다. 또한 HLA는 계층구조(layered), 자료 추상화(data abstraction), 사건기반(event-based) 아키텍처 특징을 가지고 있고, 컴포넌트 기반 실시간 분산 시뮬레이션 개발을 용이하게 해 준다(Frederick *et al*., 2000).

HLA 구성은 크게 규칙, RTI, 객체 모델 양식(Object Model Template, 이하 OMT)으로 구성되어 있으며, 시뮬레이션 사이의 상이한 하드웨어들을 통합하고, 상이한 시간 단위 및 상이한 해상도 수준의 문제를 해결해 주며, 이러한 장점으로 이미 개발된 M&S(Modeling and Simulation) 자료들의 재사용성을 보장해 준다. HLA 아키텍처에서 상호작용의 정보는 시뮬레이션들 사이에서만 일어나는 것이 아니고, 실제 참여한 객체(실 장비) 및 요소들 간, 기존에 생성되었던 자료들 사이에서도 정보교환이 일어나며, 시뮬레이션 환경의 진행사항을 파악하고자 하는 단순 관찰자들도 RTI를 통해서 정보를 획득할 수 있다.

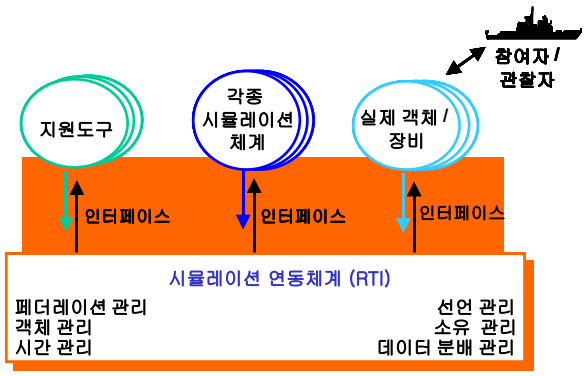

그림 1. 상위체계구조 개념도(DMSO, 1998).

2.2 HLA 구성요소

HLA는 페더레이션을 구성하는 페더레이트가 페더레이션 내에서의 상호작용을 지원하는 것이지, 페더레이션을 형성하기 위한 필요충분조건은 아니다. 따라서 시뮬레이션 구성요소 아키텍처로 HLA를 이해하는 것보다 소프트웨어 아키텍처로

이해하는 것이 더 바람직하다. HLA는 기존 시뮬레이션 자료의 송수신을 확대함으로써 모든 시뮬레이션들 간의 상호운용성을 촉진시키고, 시뮬레이션의 코드와 아키텍처 기본 틀의 재사용성을 증가시키는 것이다.

이를 위해, 각 페더레이트 간 상호운용성을 보장하기 위해서 사용하는 공통 구성요소 기반인 RTI와 페더레이트 사이의 상호관계를 명시하여, 페더레이트 간의 자료교환 및 전달방법을 정의함으로써 자료의 일관성을 유지하는 필수적인 요소인 접속규약(interface specification), 페더레이션 실행 간에 각 페더레이트의 상호작용 및 상호운용성을 보장하기 위해 준수해야 하는 객체 모델 간의 준수사항인 규칙(rules), 그리고 HLA 하 객체 모델 정보를 표현하기 위한 공통방법으로서 페더레이션과 페더레이트 객체 모델을 표현하기 위하여 표준화된 표기형식을 지정하는 OMT의 구성요소를 포함하고 있다. <그림 2>는 HLA 구성요소를 나타내는 요소이다(DMSO, 1998).

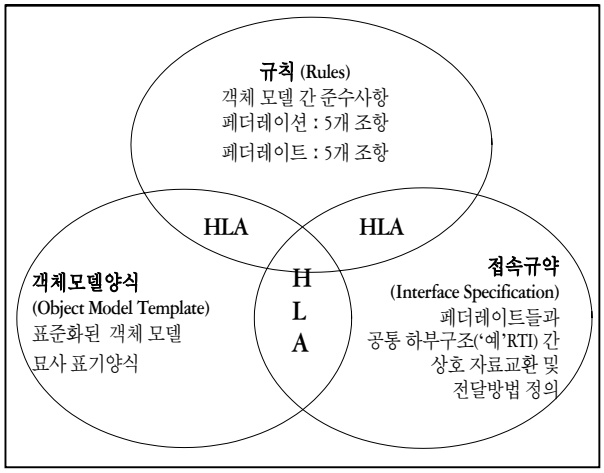

그림 2. HLA 구성요소(DMSO, 1998).

3. 컨테이너 터미널 페더레이션 목표 및 개념 모델 개발

3.1 페더레이션 요구사항 식별

컨테이너 터미널(CT) 시스템은 <그림 3>과 같이 크게 선박이 접안하는 안벽과 그 위에 설치된 C/C, 컨테이너를 적재할 수 있는(CY; Container Yard), CY 상에서 컨테이너를 처리하는 하역장비, 안벽과 CY 사이에 컨테이너의 이송을 담당하는 야드 트랙터로 이루어져 있다. 또한 시스템의 운영개념은 선박이 입항하면 운영자는 유휴선석과 C/C를 선박별 양・적하 작업량(lifts per call, 이하 LPC)에 맞게 할당하고 선박이 안벽에 접안을 하면 C/C에 의해 작업이 이루어진다. 이러한 하역과정을 모의하기 위한 컨테이너 터미널 시뮬레이션 시험기반(test bed) 체계를 고려할 수 있는데, 이 체계의 구성은 시뮬레이션 전반을 통제하는 운영통제기, 선박을 생성하는 선박생성기, C/C를 할당하는 C/C 할당모의기, 선석을 할당하는 선석 할당모의기, 그리고 하역작업을 모의하는 작업모의기로 구성될 수 있고, 각각은 통신 케이블로 연결되어 데이터를 송・수신하면서 컨테이너 터미널의 하역작업을 모의하게 된다.

이에 따른 페더레이션 요구는 시뮬레이션 시험기반 체계를 네트워크 환경하에서 PC급으로 구성하여 페더레이션을 구성하여, 컨테이너 터미널 분산 시뮬레이션을 시험할 수 있도록 한다.

3.2 페더레이션 목표

페더레이션 목표는 NT 기반 네트워크 환경에서 PC급으로 컨테이너 터미널 시스템을 구성하여 페더레이션을 구성한 후에 분산실행을 통하여 보다 현실적인 컨테이너 터미널 모의실험이 가능하도록 하고 차후 컨테이너 터미널 시스템 개발을 위한 시험기반으로 발전시키는 데 있다.

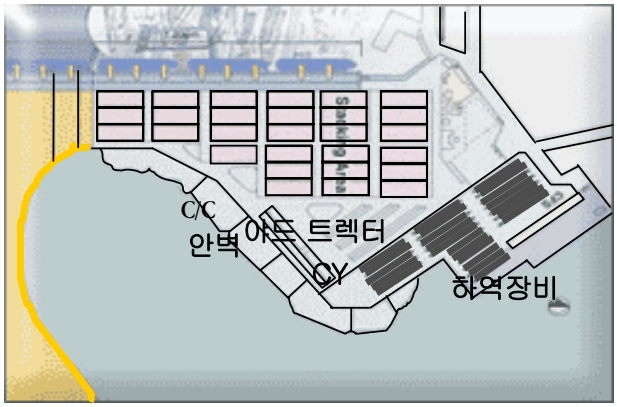

그림 3. 컨테이너 터미널 시스템.

컨테이너 터미널 페더레이션 실현에서는 <그림 4>와 같이 페더레이션을 구성하고 HLA/RTI 기반으로 각 페더레이트를 개발한다. 컨테이너 터미널 페더레이션은 SMCC(운영통제기), VEL(선박생성기), CCA(C/C할당기), PA(선석할당기), WORK(작업모의기) 페더레이트들로 구성된다.

각 페더레이트는 RTI Ambassador를 거쳐 RTI를 통하여 상호작용하게 된다. FOM은 페더레이트 상호간에 교환될 객체와 상호작용을 구성하고, 관련된 객체정보를 사용하여 각 페더레이트가 페더레이션에게 제공하는 고유한 능력을 기술하는 시뮬레이션 객체 모델(Simulation Object Model, 이하 SOM) 및 관련 클래스로 구현한다.

3.3 가정사항

HLA/RTI 하에서 컨테이너 터미널을 분석하기 위한 시뮬레

이션 모델에서는 안벽 길이, 선박의 도착분포, 선박별 LPC 분포, C/C 투입대수 및 생산성 등 입력조건의 다양한 변화에 대해 안벽에서 처리 가능한 적정 물동량의 변화를 분석한다. 이를 위한 가정사항은 다음과 같다.

① 모델 구성은 신선대(PECT) 항을 기준
② 분석을 위해 사용되는 외생적 입력 자료는 안벽능력에 가장 영향을 많이 미치는 선박별 LPC, 선박별 규모에 따라 할당되는 C/C 대수, 안벽의 길이(선석 수)로 판단
③ 특정 평가지표에 대한 목표 서비스 수준 결정
- 선박 대기시간 비율 10% 적용
- 선박 대기 비율 1% 적용

※ 선박 대기시간 비율 = 평균 대기 시간 / 평균 작업 시간

※ 선박 대기 비율 = 대기 선박 수 / 총 도착 선박 수

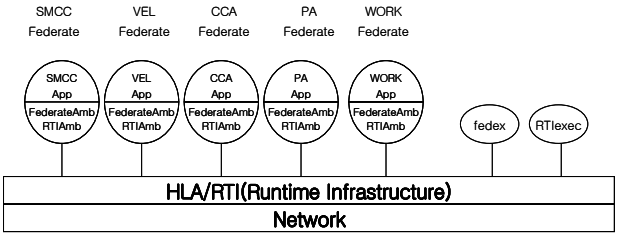

그림 4. 컨테이너 터미널 페더레이션 설계.

④ 입항선박의 하역물량은 선박에 배정된 각 C/C별로 균일하게 배분, 따라서 C/C들의 작업완료 시점은 동일하며 선박의 작업완료 시 투입된 모든 C/C는 유휴상태가 된다.
⑤ 선박의 이 · 접안 준비시간은 항상 일정
(본 연구에서는 각 90분으로 설정)
⑥ 안벽 작업시간에서 불가피한 지연, 식사시간 등의 변동요인은 배제하고 하역 준비시간은 동일하게 간주한다. 따라서 접안선박에 대한 총 작업시간당 C/C 생산성은 일정하다고 가정하고 장비의 고장이나 유지/보수는 고려하지 않는다.

3.4 시나리오 개발

안벽능력 분석을 위한 시뮬레이션 모델의 공간적 범주는 모델의 단순화를 위해 선박이 도착하는 묘박지에서 접안하여 양 · 적하 작업을 수행하는 안벽으로 제한하고, 터미널의 장치장 및 게이트에서 발생하는 하역작업들은 배제하였다. 그리고 시뮬레이션 모델의 시간적 범위는 선박이 터미널에 도착한 순간부터 출항하는 시점까지로 한정하였다. 따라서 본 연구는 안벽에서의 작업과정을 중심으로 안벽능력을 분석하는 것으로서, 장치장에서의 재고량 변화 및 게이트에서의 외부 트럭 도착 등은 시뮬레이션 모델에 포함하지 않았다.

먼저, 안벽능력을 분석하기 위해서 필요한 최소한의 기초자료에 따라 시뮬레이션 모델에서 분석하고자 하는 시나리오를 설정한다. 시뮬레이션 수행을 위해서는 입력자료로서 선박 도착분포, 안벽길이 또는 선석 수, C/C 대수, C/C 생산성, 선박의 길이 및 LPC, 그리고 선박별 C/C 할당정책 등이 설정되어야 한다.

선박 도착분포는 PECT 항을 기준으로 하였고, 1999년 1월 1일부터 12월 31일까지 도착한 총 1,113척의 선박 중 자료 분석 과정에서 작업시간이 음수이거나 48시간(2일) 이상인 선박 등 자료상의 이상치를 오류로 보고 이 자료를 제외한 902척의 입항선박에 대해서 선박 도착시간 간격분포를 분포를 추정하는 소프트웨어인 Expert-Fit를 이용하여 분석하였다. 분석결과 식 (1)과 같이 α : 1.183832, β : 8.176475인 감마분포가 PECT 항의 선박 도착분포를 가장 잘 설명하는 것으로 분석되었다.

$$f(x) = \begin{cases} \dfrac{(x-\gamma)^{\alpha-1}}{\beta^{\alpha}\Gamma(\alpha)} exp[\dfrac{-(x-\gamma)}{\beta}], & if \;\; x > \gamma \\ 0, & otherwise \end{cases} \quad (1)$$

여기서, α : shape parameter(=1.183832)
$\Gamma(1.1838382) = 0.923728$
β : scale parameter(=8.176475)
γ : location parameter(=0.019998)

안벽능력을 산정하는 방법에 있어 시뮬레이션을 적용함으로써 얻어지는 여러 이점 중의 하나는 터미널의 특성들을 상세히 모델에 반영함으로써 이에 따른 영향들을 효과적으로 분석할 수 있다는 점이다. 터미널에 입항하는 선박의 규모는 안벽의 운영과 밀접한 관련이 있다. 따라서 본 연구에서는 양 · 적하 작업시간에 차이를 보이는 입항선박의 특성을 반영하기 위해 선박들을 몇 개의 선형으로 구분하여 생성함으로써 선박의 속성을 구별하고자 한다. 터미널에 도착하는 선박의 속성을 구분하기 위하여 요구되는 정보는 선박길이, LPC 등이 있다. 선박길이는 안벽에서의 점유길이를 고려하기 위함이고, LPC는 접안선박이 안벽에서의 양 · 적하 작업시간을 구하기 위한 정보이다. 물론 양 · 적하 작업시간은 C/C 대수 및 생산성을 동시에 고려하여 계산된다.

본 연구에서는 이러한 선박의 길이 및 LPC는 선박이 도착할 때마다 결정하는 것으로 하였고 1999년도의 운영 실적자료를 입력자료로 분석하여 <표 1>과 같이 3개의 선형으로 구분하였다.

컨테이너 터미널의 선석할당은 터미널에 도착한 선박을 안벽의 어느 위치에 접안시킬 것인가를 결정하는 문제이다. 일반적으로 컨테이너 터미널을 운영할 때에는 선석계획(berth planning)에 의하여 결정한다. 선석계획 수립 시 계획기간 동안 도착예정인 모든 선박에 대해 접안위치를 결정할 때에는 선박별 LPC, 각 선박에 적하될 컨테이너가 장치된 장치장 위치, 각 선박으로부터 양하된 컨테이너가 장치될 장치장 위치, 안벽에서의 컨테이너 운영정책 등의 요소를 동시에 고려한다. 흔히

자원할당 문제(resource allocation problem)로 알려진 선석할당 문제는 주어진 정보의 종류에 따라 또는 추구하는 목적에 따라 수리모형을 다양하게 설정하여 목표계획법(goal programming) 등으로 계산하는 것이 일반적이다.

표 1. 선형(LPC) 구분

선형	LPC(lifts)			선박길이(m)		구성비율 (%)
	최대	최소	평균	최소	최대	
A	2,000	1,750	1,875			11
B	1,750	800	1,275	150	300	15
C	800	400	600			74

그러나 본 연구에서 구축하는 시뮬레이션 모델에서는 선석할당을 자원할당 수리모형에 의하지 않고, 계속 발생하는 사건들, 즉 도착한 선박이나 접안중인 선박이 작업을 완료하고 떠나는 사건이 발생할 때마다 대기중인 선박 중에서 접안할 위치를 결정하는 문제로 정의한다. 즉, 대기중인 선박이 있는가? 또는 유휴중인 선석이 존재하는가? 등을 계속적으로 판단하여 선석을 다시 할당하여야 한다. 그 과정은 <그림 5>와 같다.

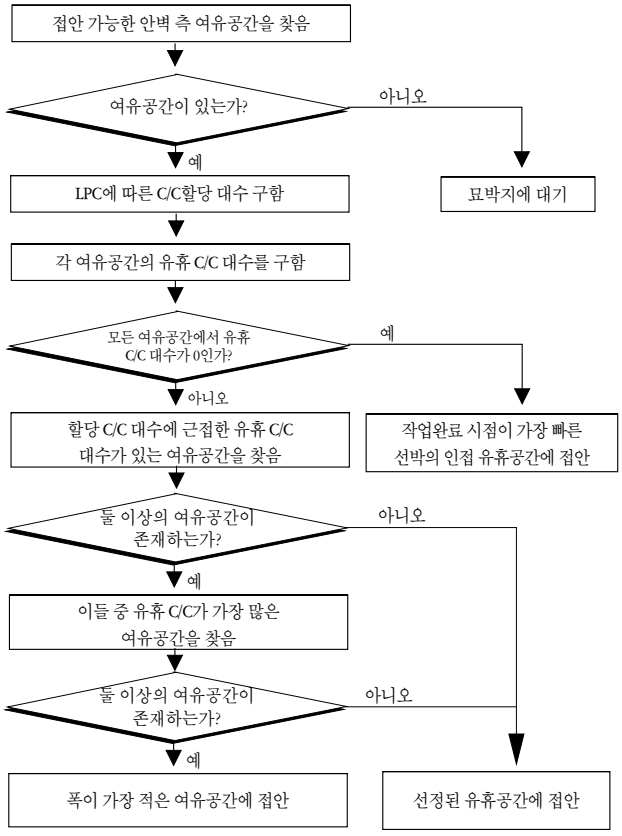

그림 5. 접안위치 결정과정.

우선순위 및 규칙은 다음과 같다.

① 접안 가능한 여유공간을 찾음
② 접안 가능한 여유공간이 없으면 접안 대기
③ 접안을 고려중인 선박 i에 할당해야 할 C/C 대수 QC_i 계산
④ 접안 가능한 모든 여유공간의 유휴 C/C 대수가 0이면 작업완료가 가장 빨리 예상되는 인근 위치에 접안
⑤ ΔQC가 0, +, −의 순서로 접안 가능한 여유공간을 찾음
$\Delta QC = | QC_i - FQ_j |$
여기서, FQ_i : 여유 공간에서 작업 가능한 유휴 C/C 대수
⑥ 동률의 ΔQC가 존재하면 여유공간이 적은 공간에 접안

선박별 C/C 할당문제는 선박의 접안시점과 이안시점에 결정해야 할 사항이다. 접안선박에 대해서는 유휴 C/C 중 몇 대의 C/C를 할당할 것인가를 결정해야 하고, 이안시점에서는 이안선박에 할당되어 작업을 수행한 C/C를 현재 작업중인 선박 중 어느 선박에 할당할 것인지 또는 대기시킬 것인지를 결정해야 한다. 이러한 C/C 운영전략은 <표 2>와 같이 각 선박별 LPC에 따라 정해져 있으며 선박의 이안사건이 발생할 때마다 C/C 할당을 재검토한다.

표 2. C/C 할당정책

선 형	C/C 할당 대수	
	최 소	최 대
A	4	5
B	3	4
C	2	3

이러한 C/C 할당정책에 따른 의사결정 과정은 <그림 6>과 같다.

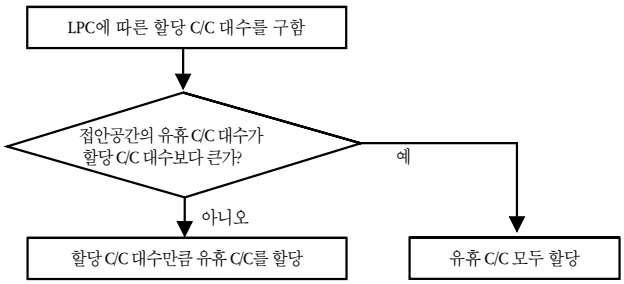

그림 6. C/C 할당과정.

컨테이너 터미널 페더레이션 요구영역에서 활동할 주요 페더레이트로서 SMCC, VEL, CCA, PA, WORK를 식별하였다. 시나리오에 따른 각 페더레이트의 임무와 상호작용은 다음과 같이 식별할 수 있다.

SMCC는 VEL에 시뮬레이션 시작 메시지를 송신한다. VEL은

SMCC로부터 시작 메시지를 받는 순간부터 부산의 PECT 항을 모델로 한 α : 1.183832, β : 8.176475인 감마분포로 선박을 생성하고 난수를 적용하여 3 가지 종류의 LPC로 선박을 구분 CCA와 PA에 선박에 대한 데이터를 실시간으로 송신한다. CCA에서는 유휴 C/C 대수를 파악하여 VEL로부터 수신받은 선박의 특성(LPC)에 따른 투입 C/C 대수를 판단 WORK로 자료를 송신한다. PA는 유휴 선석 수를 판단, 유휴 선석 유무를 WORK로 송신하고, WORK는 CCA, PA로부터 C/C와 선석이 작업 가능하다면 작업을 실시하고, 한 가지 조건이라도 맞지 않으면 선박을 대기시킨 후 그 결과를 SMCC에 송신한다.

이와 같이 분석된 임무와 상호작용에 따라 UML을 지원하는 비주얼 모델링 툴인 Rational Rose 2000을 사용하여 모델링을 수행할 수 있다. <그림 7>은 사용 사례도를 보여준다.

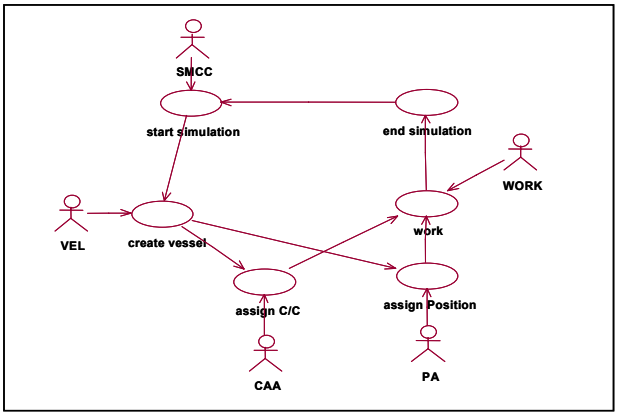

그림 7. 사용 사례도(Use case diagram).

각 행위자(actor)는 각 페더레이트 사용자로서 표현되어 있으며 중요 수행업무를 모델링하였다. 먼저 SMCC는 페더레이션을 생성하고 시뮬레이션을 시작하며 각각의 페더레이트 운영자들은 이 페더레이션에 참여하고 각 객체를 생성시킨다.

VEL은 선박 객체를 생성시켜 초기화하고 그 특성을 부여한다. CCA는 유휴 C/C를 판단하여 C/C를 할당하고 PA는 유휴 선석을 판단하여 선석을 할당한다. WORK는 할당된 C/C와 선석으로 양・적하 작업을 모의하고 시뮬레이션 종료조건(기간)을 만족하면 SMCC는 시뮬레이션을 종료한다.

사용 사례도를 통하여 각 페더레이트들의 중요 수행함수와 객체를 식별하여 모델링하였다. 다시 이를 세분화하여 각 페더레이트들의 클래스를 모델링한다. 5개의 페더레이트를 각각 분할하여 해당되는 주요 클래스들을 모델링할 수 있다. 전체의 클래스 집합은 하나의 프로그램이 아닌 페더레이션을 표현하고 있으며, 다시 이를 페더레이트별로 분할하여 클래스들을 세분화하고 모델링한다. <그림 8>은 클래스들을 모델링한 것을 나타내는 클래스도이며 HLA가 제공하는 FedAmbassador 클래스로부터 상속받은 클래스들(SMCCFedAmb, VELFedAmb, CCAFedAmb, PAFedAmb, WORKFedAmb)들이 각각의 페더레이트를 지원하고 RTIAmbassador가 페더레이트들에게 RTI 서비스

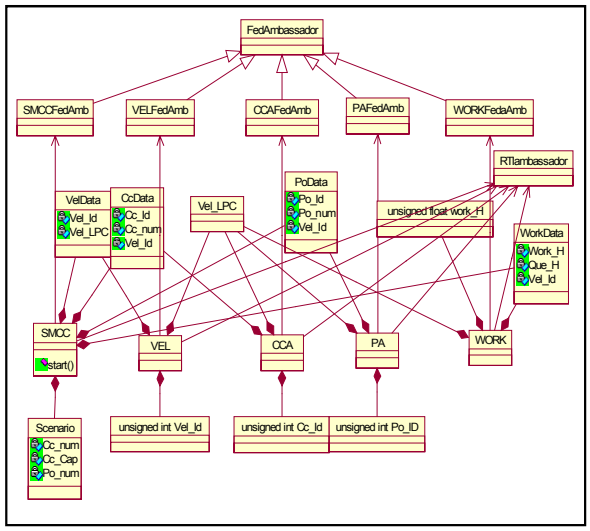

그림 8. 클래스도(Class diagram).

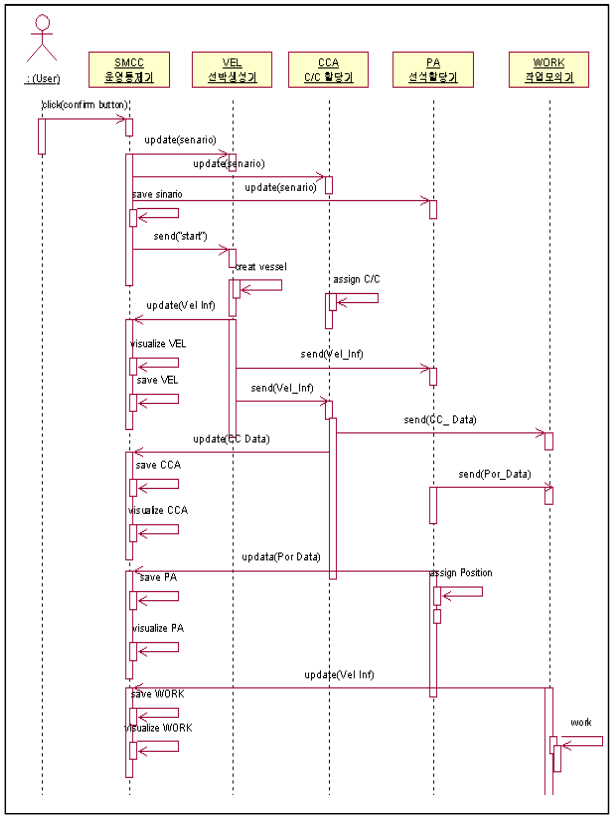

그림 9. 순서도(Sequence diagram).

를 지원하는 관계 등을 설명하고 있다.

이러한 클래스도를 바탕으로 페더레이트들 간의 시간에 따른 순서도는 <그림 9>와 같이 식별할 수 있다. <그림 9>에서 시간에 따른 각 페더레이트의 임무수행과 상호작용을 알 수 있는데 먼저 사용자가 SMCC 페더레이트에 페더레이션 시작을 명령하면 SMCC 페더레이트는 시나리오를 생성하여 각각의 페

더레이트에 전파한다. 시나리오를 받은 VEL 페더레이트는 선박을 생성하고 공표하여 다른 페더레이션의 요청에 따라 선박정보가 이용 가능하게 한다. CCA 페더레이트와 PA 페더레이트는 각각 선박정보에 따라 C/C 및 선석을 할당하여 그 정보를 공표한다. WORK는 공표된 정보들을 종합하여 작업을 모의하는 순차적 과정 또한 알 수 있다.

이와 같이 UML을 통하여 전체의 사용 사례를 모델링하여 분석하고, 이를 통하여 각 페더레이트에서 생성해야 할 클래스들을 모델링하였다.

3.5 페더레이션 구성

페더레이션 구성은 <그림 10>과 같이 LAN 상에서 Window 98 환경하에 운영통제기(SMCC), 선박생성기(VEL), C/C 할당기(CCA), 선석할당기(PA), 작업모의기(WORK)로 구성한다.

페더레이션 기능할당은 SMCC가 시뮬레이션의 진행을 위하여 가상영역인 페더레이션을 생성하는 역할을 수행하고 시뮬레이션이 모두 종료되면 페더레이션을 소멸시키는 역할을 수행하며, VEL은 항만에 도착하는 선박을 생성, 선박의 선형을 결정하는 역할을 수행한다. CCA는 선박생성기에서 생성된 선박의 선형에 따라 C/C를 할당하는 역할을 담당하고 PA는 VEL에 의해 생성된 선박에 선석을 할당하는 역할을 맡으며 WORK는 CCA와 PA에서 C/C와 선석이 할당된 후 항만에서의 작업을 모의하는 역할을 수행한다.

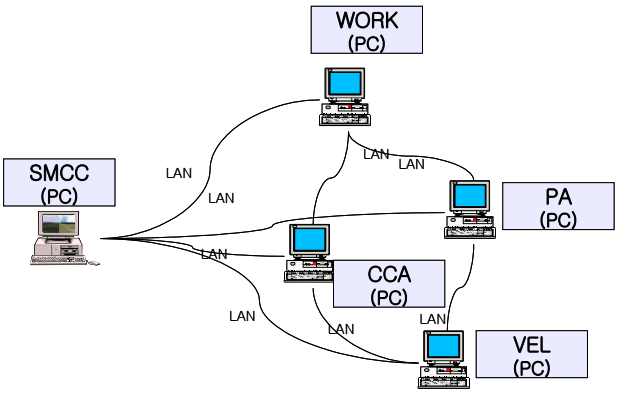

그림 10. 페더레이션 구성.

4. 컨테이너 터미널 페더레이션 개발

4.1 FOM 개발

FOM을 개발하는 목적은 페더레이션 내의 모든 페더레이트 간에 공통적으로 사용될 객체와 기타 산물들을 규정하기 위함이다. HLA는 모든 FOM의 구조를 규정하고, 각 페더레이션은 고유한 표현수단인 FOM을 갖고 있다. 각 페더레이트의 실행은 페더레이트의 SOM을 이용하여 페더레이트가 서로 대화하는 것과 사건의 이름을 정의한다. FOM은 페더레이션 실행 동안 RTI를 통해 변경되는 자료를 표현하는 수단이다. FOM은 실행이 시작될 때 RTI에 자료로 공급된다는 관점에서 RTI에 대한 매개변수이며, RTI는 FOM이 변경되거나 RTI가 다른 페더레이트에 적용될 때에는 변하지 않는다. FOM의 주요 구성요소는 객체 클래스와 상호작용 클래스이다.

각 페더레이트의 설계자는 RTI나 HLA 기준에 변경 없이 자신의 객체 모델을 채택하는 데 자유롭다. 일반적인 FOM 개발방법은 이상적인 페더레이션 FOM을 포함하는 페더레이션 개념모델과 객체 모델 자료사전(OMDD; Object Model Data Dictionary)을 사용하여 상향식 페더레이션 FOM을 구축하는 방법, 참여하고 있는 모든 페더레이트 SOM을 함께 병합하는 방법, 문제영역을 완전하게 나타낼 수 있도록 다른 페더레이트 SOM들을 합하는 방법, 이전의 유사한 응용 프로그램으로부터 페더레이션 FOM을 시작하여 수정 및 향상시키는 방법, 그리고 사용자에게 잘 알려진 참조 공통구도를 제공할 수 있는 FOM에서 시작하여 응용 프로그램에 필요하지 않는 요소를 제거하고 필요에 따라 수정 보완하는 방법 등이 있다(DoD, 1998).

본 연구에서 사용한 FOM의 개발도구는 미 국방성 모델링/시뮬레이션 본부(DMSO)에서 개발한 최신 버전인 OMDT Ver. 1.3이고, 상향식 페더레이션 FOM 구축 중심으로 개발하였다. 그 결과는 <그림 11>과 같다.

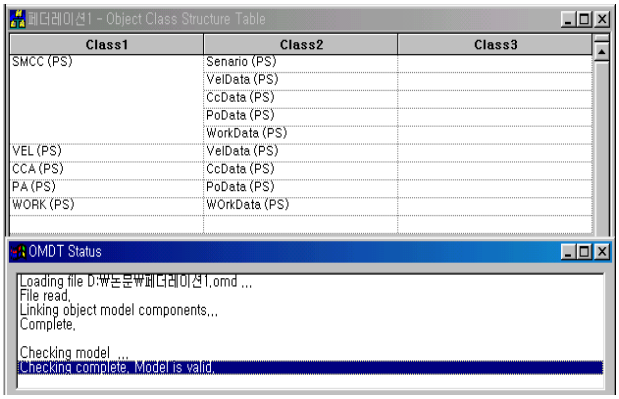

그림 11. OMDT를 이용한 FOM 개발.

4.2 RTI 서비스 식별

페더레이션을 구현하는 데에 <그림 1>에서 보는 것과 같이 가용한 6가지 RTI 서비스가 제공된다. 즉, 페더레이션 관리 서비스, 선언관리 서비스, 객체관리 서비스, 소유권관리 서비스, 시간관리 서비스 및 분배관리 서비스가 제공된다. 이들 서비스 중에서 컨테이너 터미널 페더레이션 구현을 위해서는 페더레이션관리 서비스, 선언관리 서비스, 객체관리 서비스, 그리고 시간관리 서비스 기능을 사용한다(DMSO, 1998).

구체적으로 페더레이션관리 서비스를 통하여 페더레이션

생성을 RTI에게 요구하면 RTI가 실행되고 있는 컴퓨터의 IP 번호를 근간으로 페더레이션을 생성한 후, FedExe를 실행시키고 자신도 참여하게 된다. 그리고 교환 및 관련 서비스는 FOM의 Fed 파일과 연계하여 이루어지는데, 예를 들어 서비스 구현에서 초기에 fed 파일을 이용하여 기 모델링된 내용들을 사전에 파싱(parsing)을 실시하고 서비스를 시작할 준비를 한다. 선언관리 서비스에서는 미리 준비된 메모리에 각 페더레이트는 해당 RTTS(RunTime Type String)를 선언하고 RTI부터 할당된 핸들(handle) 번호를 받아간다. 받아온 핸들 값을 저장하는 임시변수를 RTTI (RunTime Type Identification data)에 저장하여 사용한다. RTI는 핸들 값을 생성하고 나서 각 주소를 기억하고 있다가 동일한 객체유형의 핸들 번호끼리 상호공표(publish)와 요청(subscribe)을 실시하면 가상적인 통신 채널을 형성시켜 상호간 속성 및 상호작용 갱신을 주선한다.

객체관리 서비스에서는 데이터 송・수신이 이루어지며 객체의 데이터 송・수신을 일방의 페더레이트가 Register()를 한 후 RTI가 반대편의 페더레이트에서 discoverObjectInstance()를 호출하면서부터 본격적인 서비스를 시작한다. 수신하는 데이터는 페더레이트 내에 메모리를 할당하여 저장한다. 이때 모든 컴퓨터는 내부적으로 고유한 바이트(byte) 전환방법을 가지고 있으므로 컴퓨터 간의 통신에는 반드시 그 전환을 통일시켜 주어야 한다.

또한 각 페더레이트는 자신의 객체가 아닌 다른 객체를 수신하기 위해서 수신 모듈을 구현해야 한다. 수신 모듈은 여러 가지 방법이 가능하지만 RTI를 통해서 각 객체의 핸들과 속성 및 파라미터를 분배(sorting)하는 방법을 고려하여 모듈을 구현한다.

핸들 값을 받아내기 위해 먼저 헤더 파일에서 선언한 다음 cpp파일에서는 기존의 Fed 파일에서 모델링한 것과 동일한 내용으로 초기값을 선언한다. 페더레이트가 RTI에 접근할 때 먼저 Fed 파일을 RTI에 제시함으로써 RTI는 Fed 파일의 내용대로 파싱을 실시한다. 파싱된 내용을 근거로 RTI는 페더레이트에서 요구하는 핸들 값을 생성하여 각 페더레이트들에게 넘겨주게 된다.

4.3 페더레이션/페더레이트 설계

페더레이션 내에서 페더레이트 간 객체의 데이터 교환을 실시하려면 일종의 수신단이 필요하다. 동일한 페더레이트가 상호 데이터를 교환할 때는 별 문제가 없지만 그 성격이 아주 다른 이종의 페더레이트가 페더레이션에 참가하여 진행하려면 다른 종류의 페더레이트가 보내는 데이터를 받을 수 있도록 구조를 새로이 형성시켜 주어야 한다. 이러한 방법에는 여러 가지 방안이 가능하지만 객체지향적으로 페더레이션을 완성하도록 페더레이트 구성방안을 연구하였다. 본 연구에서 개발한 방법은 <표 3>과 같이 모두 3가지 방안으로 구성하였다.

FederateAmbassador는 RTI로부터 메시지 인보케이션에 의해 해당임무를 수행하며 RTI 서비스를 해당 페더레이트에게 전달하는 역할을 수행한다.

CTFederateAmbassador 클래스는 FederateAmbassador 클래스로부터 상속받아 사용자가 필요한 형태로 변형하여 클래스를 작성하여 페더레이트와 인터페이스시킨다.

CTFederateAmbassador 클래스에 RTI로부터 DiscoverObject-Instance( )가 호출되면, 해당 객체인지를 먼저 확인한 후, 페더레이트 내에 가상의 객체를 형성시킨다. 형성된 가상의 객체를 통해서 페더레이트는 RTI에 프로세스 진행에 필요한 속성들을 요청하고 해당 속성들이 상대편에서 갱신(update)될 때마다 RTI는 CT FederateAmbassador에 reflect( )를 호출한다. 호출된 함수의 루틴 내에서 해당 객체가 확인되면 시간관리 정책에 따라 자체적으로 받은 데이터를 이용하여 갱신하고 다시 자체 프로세스를 진행한다. 이런 식으로 CTFederateAmbassador가 직접적으로 페더레이트 내부의 프로세스에 전반적으로 참여하는 방법을 전역적 가시성(global visibility)을 가졌다고 정의한다. 이 방법은 페더레이트 내부의 프로세스가 RTI의 영향을 직접적으로 받기 때문에 순수하게 객체지향적인 방법으로 페더레이트를 디자인하기 곤란하다.

표 3. 페더레이트 구조설계 방법별 특징

구분	구현방법	적용
클래스 간 객체/속성 교환	Class 대입방법	단순한 프로그램
	Ghosting 방법	복잡한 프로그램
다른 클래스와 객체/속성 교환	Attribute 방법	소규모

위의 전역적 가시성과는 달리 CTFederate- Ambassador 클래스로부터 한번 더 상속 받은 임의의 FedAmbassador 클래스를 생성하고, 생성 시 페더레이트의 객체 생성주소를 FedAmbassador 생성 시 함께 생성시켜, main문에서 객체생성 시, 이 주소값을 이용하여 바로 사용하도록 한다. 따라서 객체가 CTFederate Ambassador의 사건기반에만 의존하여 프로세스를 진행하는 것을 차단하고, 불필요하게 페더레이트의 프로세스에 관여하지 않도록 할 수 있다. 따라서 페더레이트 자체는 좀더 객체지향적인 개념의 디자인을 수행할 수 있다. 이러한 설계 디자인을 지역 가시성(local visibility)을 가진다고 정의한다. 지역 가시성은 페더레이트와 FederateAmbassador의 연관성 측면에서 얼마나 객체지향적 구조로 디자인되었는가에 대한 관점이며, 차후 HLA 구현상에 있어서 중요한 논점이 될 것으로 예상된다. 이는 곧 HLA 구현이 얼마나 객체지향적인 요소를 가질 수가 있겠는가에 대한 논의가 될 수 있으며, HLA 근본 목적인 재사용성과 상호운용성 확보에 관건이라고 할 수 있다.

HLA는 객체지향적인 구조가 아니더라도 구현할 수 있다. 그러나 페더레이션 내에서 운용될 객체 수와 속성 및 상호작용과 파라미터가 많을 때에는 객체지향적인 모듈화 작업은 필수

적이라고 할 수 있다. 컨테이너 터미널 페더레이트들에는 지역가시성을 가지도록 설계하였다.

컨테이너 터미널 페더레이션에서는 SMCC, VEL, CCA, PA, WORK의 모두 5개의 페더레이트로 구성된다. 앞에서 살펴본 설계 패턴 중에서 ghosting 방법과 attribute 방법을 적절하게 사용하여 페더레이션을 구성하였다.

선박의 정보를 생성하여 클래스화하는 페더레이터는 VEL 페더레이트이고, 선박의 정보를 필요로 하는 클래스들은 CCA, PA, WORK 페더레이트들이다. 이러한 페더레이트 간의 인터페이스는 ghosting 방법으로 설계하여 메시지를 송・수신하도록 계획하였다. 또한 WORK 페더레이트는 CCA 페더레이트에서 공표하는 C/C의 할당 정보와 PA 페더레이트에서 공표하는 선석 할당 정보가 추가적으로 더 필요한 페더레이트이다. 이것 또한 ghosting 모듈을 삽입하면 모든 정보를 획득할 수 있으나 객체지향적인 관점에서의 설계를 위해 attribute 방법을 이용하여 모든 정보를 동시에 송신하도록 설계하였다.

5. 페더레이션 통합

5.1 페더레이션 통합 구현

페더레이션 통합 개발을 위해서 각 페더레이트는 할당된 기능에 대하여 구현하고 페더레이션과 인터페이스는 FOM을 근간으로 RTI 서비스를 사용하여 구현한다. 그리고 각 페더레이트를 RTI 상에서 통합 실현하여 개발한다. 다수 팀이 개발할 경우에는 개발참여 시 각 페더레이트별로 필요한 공표, 요구들을 테이블화하여 각 페더레이트별로 동시적으로 개발할 수 있지만, 단일팀이 개발 시에는 가장 많이 사용될 객체를 포함하는 VEL부터 개발하여 페더레이트별로 검증을 거치면서 통합개발을 이루어 갈 수 있다. 본 연구에서는 후자 방식대로 개발하였다.

5.2 객체지향적 구현

기존의 객체지향적 방법을 HLA 모델에 적용하기 위하여 먼저 HLA 구조의 특성과 객체지향 기법 간의 차이를 구분 식별하여 보면, 기존의 객체지향적인 프로그램은 단순히 현실 객체를 추상화하여 클래스로 구조화한 후 이를 인스턴트(instant)로 생성하여 구현함으로써 현실 세계와 프로그램 상의 객체와의 의미적 차이를 감소시킬 수 있으며, 프로그램의 직관성, 재사용성, 유지 및 보수성이 용이해 질 수 있다. 객체지향에서는 그 단위가 객체와 메시지로써 클래스의 상속, 포함, 연관관계로써 구현된다.

HLA에 이러한 객체지향적 프로그램을 실시하려면 상기와 같은 요소가 식별되어야 한다. 즉, HLA에서는 객체와 상호작용으로 클래스를 구분하여 분산 네트워크 상에서 데이터 관리 기능을 강화하는 구조로 설계되어야 하며, 이는 RTI 서비스에 의해서 지원받을 수 있다.

5.3 페더레이션 시험

페더레이션 개발에 대한 시험을 하기 위해서는 각 페더레이트별로 자체 페더레이션 생성 가능 여부를 검사하고 페더레이션 실행 간에 의도된 바대로 교환이 적절히 이루어지는가를 검사하도록 한다.

5.4 페더레이션 실행

페더레이션 구현은 마이크로소프트사에서 제작한 프로그램 제작 툴인 Visual C++ 6.0을 이용하여 C++ 언어를 기반으로 개발하였고, NT 상에서 RTI 1.3NG-V3.2를 미들웨어로 사용하였다. RTI 1.3NG- V3.2는 구버전의 RTI 소프트웨어와는 달리 콘솔(consol) 모드를 따로 분리하여, RTI에 접속중인 각 페더레이트의 상태와 이름, 핸들(handle) 번호 등을 콘솔을 통하여 확인할 수 있도록 지원하고 있다. RTI 콘솔 전시화면을 통하여 페더레이션 관리자가 명령어를 사용하여 페더레이션 멤버 정보를 알 수 있다.

최초에 RTIEXE을 시작시키면 <그림 12>와 같이 초기화면이 나타난다. 이러한 RTI 초기상태에서 페더레이션을 생성할 수 있는 명령을 실시하면 RTIEXE 프로세스는 FEDEX 프로세스를 구동(launching)하고 페더레이션이 가상공간 내 유일할 수 있도록 중요 정보를 포함한다.

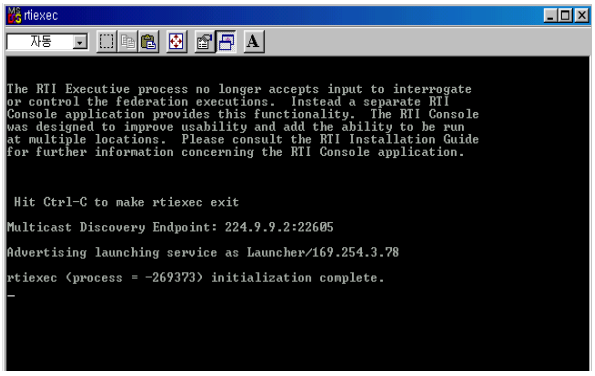

그림 12. RTI 초기화.

우선 RTI를 초기화하고 페더레이션을 생성하여 RTIEXE 프로세스가 FEDEX 프로세스를 구동(launching)함을 확인하고 각각의 페더레이트를 실행시켜 페더레이션에 참가함을 확인하여 HLA/RTI의 재사용성과 상호운용성을 입증한다. 즉, 하나의 컴퓨터에서 실행된 페더레이션에 각각의 컴퓨터에서 구현된 페더레이트들이 RTI를 통하여 페더레이션에 참가하고, 서로 연동되는 모습을 통하여 상호운용성을 확인하고 각 페더레이트에서 구현된 객체들이 RTI에서 공표하고 요청하여 다른 페

더레이트에서 자유롭게 객체를 사용하는 것을 확인함으로써 재사용성을 입증한다.

페더레이션을 생성하면 <그림 13>과 같이 RTI 초기화 창이 rtiexec 창에서 fedex 창으로 바뀌게 되고 생성된 페더레이션의 process ID와 Endpoint를 제공하여 RTI 서비스를 실시할 수 있는 준비를 완료하게 된다.

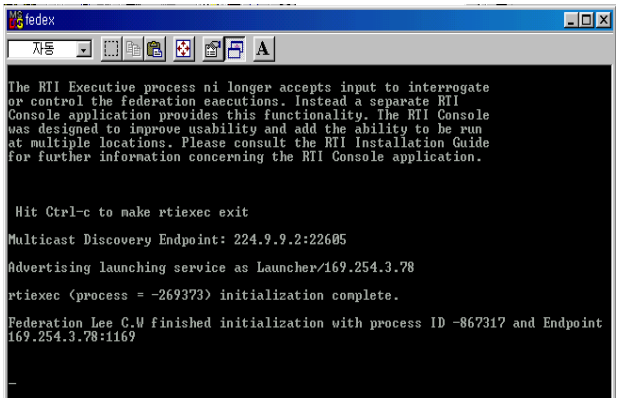

그림 13. 페더레이션 생성 시현.

SMCC(운영통제기)에서 페더레이션 생성을 요청하면, RTI는 Fed 파일의 내용에 의거 파싱(parsing)을 실시한 후 FedExe를 호출하여 페더레이션 관리를 시작한다. SMCC는 <그림 14>와 같이 페더레이션을 RTI에 요청하여 생성하고 객체를 가상공간에 등록한다.

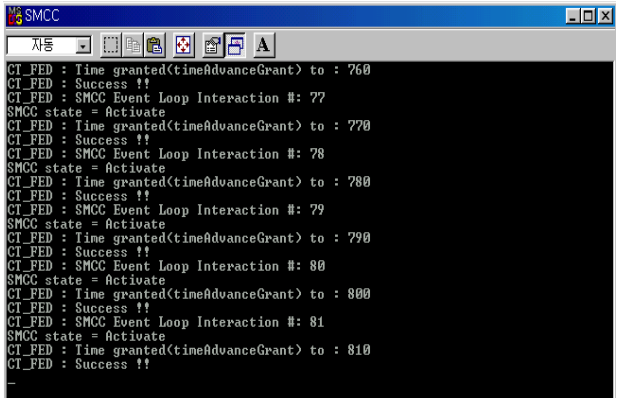

그림 14. 운영통제기(SMCC) 객체 생성.

선박생성기(VEL) 페더레이트는 페더레이션에 참가하여 선박을 생성하여 그 정보를 VelData 클래스로 페더레이션에 등록하고 공표(publish)한다. <그림 15>에서 선박생성기에서 선박을 생성하고 각각의 선박에 ID와 LPC를 지정하여 VelData 클래스로 공표함을 알 수 있다. 또한 운영통제기 페더레이트와 다른 컴퓨터에서 구현된 선박생성기 페더레이트가 RTI를 통해 서로 연동됨을 알 수 있다. 공표된 VelData 클래스는 다른 페더레이트들이 요청만하면 사용할 수 있게 된다.

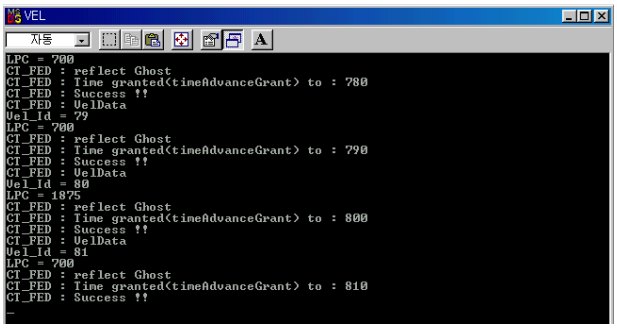

그림 15. 선박생성기(VEL)의 선박 생성.

C/C 할당기는 생성된 선박에 대하여 C/C를 할당한다. <그림 16>에서 선박생성기에서 공표된 선박정보를 요청하여 그 정보에 따라 C/C를 할당하여 CcData 클래스로 공표함을 알 수 있다.

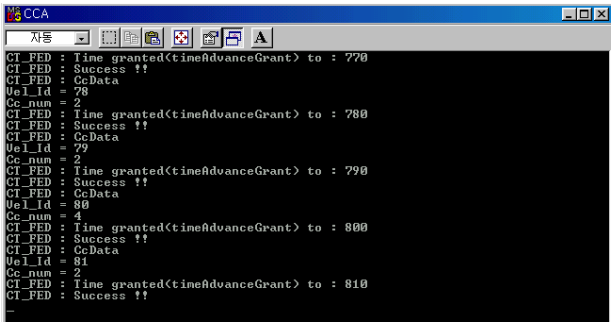

그림 16. C/C 할당기(CCA)에서의 C/C 할당.

선석할당기(PA)는 생성된 선박에 대하여 선석을 할당한다. <그림 17>에서 선박생성기에서 공표된 선박정보를 요청하여 그 정보에 따라 선석을 할당하여 PoData 클래스로 공표함을 알 수 있다.

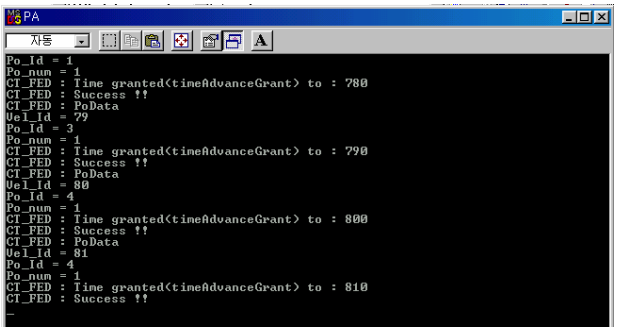

그림 17. 선석할당기(PA)에서의 선석 할당.

작업모의기는 작업을 모의한다. <그림 18>에서 선박생성기와 C/C 할당기 그리고 선석할당기에서 공표된 선박정보, 할당된 C/C 및 선석에 관한 정보를 요청하여 그 정보에 따라 작업을 모의하여 작업시간과 대기시간을 표현하는 것을 알 수 있다.

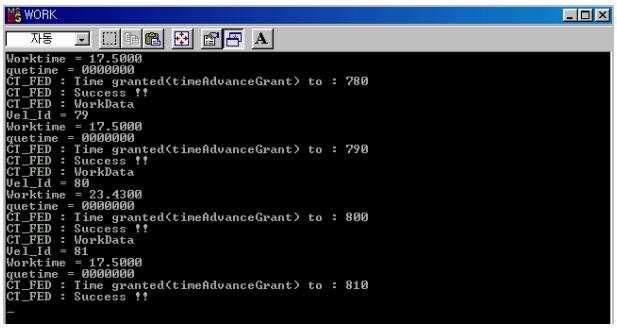

그림 18. 작업모의기(WORK)에서의 작업 모의.

그림들은 각 페더레이트가 페더레이션에 참가하는 순서별로 나열하였다. 페더레이션 구성은 RTI에 먼저 페더레이션 영역을 요청하는 운영통제기(SMCC) 페더레이트가 구성하게 되고, 나머지 페더레이트들은 이에 참가한다. 이렇게 다른 컴퓨터에서 구현된 페더레이트들이 RTI를 통해서 운영통제기가 생성한 페더레이션에 이상 없이 참가함은 HLA가 가지고 있는 상호운용성을 보여주는 것이다. 또한 각 페더레이트는 단위시간별로 프로세스(process)를 진행하며, 그 때마다 갱신된 속성들을 RTI를 경유하여 필요한 페더레이트들에게 전달하게 되는데, 이는 HLA의 재사용성을 보여주는 것이다.

6. 페더레이션 실행 및 분석

6.1 페더레이션 실행결과

본 연구에서 구축한 시뮬레이션 모델의 현실성을 확인하기 위해 4개 터미널의 운영실적 자료를 참고하였다. 모델의 확인을 위해서는 한 터미널의 운영실적 자료를 사용하여야 하나 터미널마다 자료의 형식이 다를 뿐만 아니라 또한 자료가 부족하여 부산의 신선대항(PECT)을 기준으로 4개 터미널의 자료를 일부 혼용하였다. 모델의 현실성을 분석하기 위한 입력자료로 안벽길이 1,200m(300m 선석 4개), 연간처리 물동량 120만 TEU(80만 lifts), 총 투입 C/C 대수 11대, C/C 생산성을 시간당 20 lifts로 설정하였다.

페더레이션에서 구해진 통계치 결과를 <표 4>에 정리하였다. <표 5>에서는 시뮬레이션 모델에서 수행된 결과들의 평균값과 실적 통계치를 비교하였다. 항목에서 선박별 평균작업량과 평균 연간 작업량(8주 수행값이 구해지므로 52주인 경우의 물동량으로 추정)은 입력된 실적자료의 값이 시뮬레이션 모델에서 제대로 반영되어 생성되는지를 확인하기 위한 것이다. 그리고 평균 접안시간, 평균 선석점유율(berth occupancy rate)과 평균 대기시간은 시뮬레이션 모델에 묘사된 프로세스와 운영논리가 현실과 비교하여 타당한지를 검토할 수 있는 기준으로 사용된 항목이다.

연간 물동량을 80만 lifts로 가정하여 선박도착 간격 분포를 구현하였는데 시뮬레이션 수행결과에서도 78만 9천 lifts를 처리하여 실제상황과 거의 일치함을 알 수 있다. 선박당 접안시간은 12.1시간, 시뮬레이션 결과는 13.4시간으로 1.3시간의 차이가 발생하고 있다. 그렇지만 이는 입력자료로 사용한 시간당 C/C 생산성 20 lifts가 과소평가되었다든지 또는 선박별 C/C 할당 대수가 전체적으로 적게 할당되었기 때문으로 추정된다. 그리고 A 터미널에서 발표한 선석 점유율은 42.7 %, 시뮬레이션 결과로 제시한 선석 점유율은 43.2 %로 나타나고 있으며, A 터미널에서 발표하지는 않았지만 시뮬레이션 결과 총 도착 선박 중 10.1 %의 선박이 도착 즉시 접안하지 못하고 대기하는 것을 알 수 있다.

표 4. 시뮬레이션 수행결과

실험 번호	연간 작업량 (TEU)	선박별 작업량 (시간)	선석 점유율 (%)	대기비율 (%)
1	120,120	831.2	44.2	9.6
2	118,221	842.8	45.6	9.7
3	118,232	831.1	43.2	12.5
4	117,298	821.8	42.7	7.8
5	118,378	825.2	42.9	11.1
6	121,511	853.5	42.0	8.6
7	119,787	832.4	43.6	11.4
8	118,889	831.9	44.5	7.1
9	118,191	827.2	43.7	8.9
10	116,980	824.6	42.4	12.7
11	121,291	834.6	42.3	7.6
평균	118,990	832.39	43.372	9.731

표 5. 시뮬레이션 결과와 운영실적치 비교

항 목	A터미널 실적치 (1999년)	시뮬레이션 수행결과
연간 하역량	80만 lifts	78.9만 lifts
선박당 평균 하역량	843 lifts	832.4 lifts
선박당 접안시간	12.1 시간	13.4 시간
선석 점유율	42.1 %	43.2 %
대기비율	-	10.1 %

6.2 민감도 분석

일반적으로 시뮬레이션 모델의 구성요소는 크게 모델의 구조(structure)와 입력자료(input data)로 되어 있다. 본 연구에서는 모델의 구조와 관련된 선석할당 및 C/C 배정 등의 운영논리를 변경하는 실험은 실시하지 않고 주요 입력자료를 변화시킬 경

우 일어나는 변화("what if")를 관찰하여 입력자료와 출력결과의 원인－결과 관계를 검토하여 추세나 변동이 타당한지를 분석함으로써 모델 확인에 대한 과정을 수행하였다. 민감도 분석을 위해 사용된 주요 입력자료의 요소로는 연간 처리물동량은 120만 TEU, 130만 TEU, 140만 TEU로 구분하고, 총 C/C 대수는 9대와 12대, 작업시간당 C/C 생산성은 20 lifts, 25 lifts, 30 lifts 및 35 lifts로 구분하여 24개의 시나리오에 대하여 시뮬레이션을 수행하였다.

그 결과 <표 6>과 같이 연간 140만 TEU를 목표 서비스 수준으로 처리하기 위해서는 C/C가 12대이고 C/C 생산성이 20 lifts일 때, 대기시간 비율은 3 %로 목표수준 10 %를 만족하지만 대기비율은 10 %로 목표수준 1 %를 만족시키지 못한다. 결국, 시간당 생산성이 35lifts 이상인 C/C가 12대일 때 목표수준을 만족시키는 것으로 나타나 한국해양수산개발원에서 시뮬레이션 모델을 이용하여 연구한 컨테이너 터미널 안벽능력 분석(김창곤 외, 2000)결과와 유사하다.

표 6. 시뮬레이션 결과(물동량 : 140만 TEU)

C/C (대)	C/C 생산성 (lifts/hour)	대기비율 (%)	대기시간비율 (%)
12	20	10	3
	25	4	1
	30	2	0
	35	1	0

6.3 기존 시뮬레이션과 차이점

HLA를 이용하여 페더레이션을 개발하는 것과 기존 시뮬레이션과의 차이점은 HLA의 구성요소 중 하나인 OMT를 이용하여 각각의 페더레이트를 다른 개발자들이 동시에 개발함으로써 페더레이션을 개발하는 시간을 단축시킬 수 있다는 것과 페더레이트와 페더레이션의 상호운용성 및 재사용성을 들 수 있다. 그 내용을 세부적으로 살펴보면 다음과 같다.

첫째, 개발의 용이성과 시간의 단축이다. HLA의 구성요소 중 OMT라는 공통 서식을 이용, 동일한 객체와 클래스로 페더레이션을 구성하는 각각의 페더레이트들을 각각 다른 개발자들이 개발한 후 RTI를 이용, 연동함으로써 페더레이션을 완성할 수 있다. 이때 개발 언어의 제한은 없다. 이는 같은 언어로 하나의 컴퓨터를 이용하여 한명 또는 한 팀에 의해 개발되었던 기존의 시뮬레이션에 비해 명확한 개발의 용이성이 있고 개발시간의 단축이 가능하다고 할 수 있다.

둘째, 페더레이션 및 페더레이트의 상호운용성이다. 기존의 시뮬레이션들은 서로 다른 언어나 툴로 개발된 시뮬레이션 간의 연동이나 서로 다른 분야－예를 들어 컨테이너 터미널 시스템에서 안벽에서 이루어지는 컨테이너 작업과 컨테이너 야드(CY)에서 이루어지는 수송작업－의 시뮬레이션은 서로 연동할 수 없었다. 그러나 HLA를 이용하여 페더레이션을 개발한다면 그 개발 언어에 상관없이 RTI를 이용한 상호연동이 가능하다.

셋째, 페더레이션 및 페더레이트의 재사용성이다. 기존의 시뮬레이션은 같은 분야의 시뮬레이션이라 할지라도 가설이 달라지거나 개발 언어가 달라지는 등 개발 환경의 변화에 따라 처음부터 다시 시뮬레이션을 개발하여야 하지만 HLA를 이용해서 페더레이션을 개발할 경우, 이전의 페더레이션에서 중복되는 페더레이트를 그대로 사용함으로써 모형의 확장성 및 재사용성을 용이하게 보장받을 수 있다.

이상의 기존 시뮬레이션과는 차별화되는 HLA 하에서 개발된 페더레이션의 특징을 이용한다면 보다 용이하고 절감된 비용과 단축된 시간으로 페더레이션을 개발하여 원하는 시뮬레이션을 수행할 수 있을 것이다.

7. 결 론

본 연구에서는 HLA 기반으로 실시간 분산 시뮬레이션 구현 사례를 보여주고 있다. 특히, 컨테이너 터미널 페더레이션을 대상으로 구현하여 제시하였다. 범위를 항만 시스템 중에서 안벽분야만으로 제한하였지만, HLA의 재사용성과 상호운용성을 이용한다면 항만 시스템 전체에 대한 페더레이션 개발이 가능할 것이다.

이러한 연구결과는 미 국방성을 중심으로 국방 관련 모든 시뮬레이션 개발에 HLA 표준을 준수하도록 규정하고 있고, 앞으로 시뮬레이션 아키텍처의 주도적 역할을 할 것으로 판단되는 HLA를 국내 민간산업 분야에 구현기술을 제공할 수 있을 것이다.

그러나 RTI 성능에 있어서 아직까지 추가적으로 연구해야 할 분야가 많다. 특히 체계 노드가 증가할수록 요구되는 성능이 나오는지에 관한 문제와 서로 다른 시간진행 메커니즘을 가지고 있는 시뮬레이션과 연동했을 때 현재 제공되는 RTI로 구현이 가능한가에 대한 문제가 검증되어야 할 분야이다. 연구과정에서 아직까지 RTI 성능이 개발 요구자 성능에 미치지 못할 수도 있음을 인지하였고 이러한 문제에 대한 연구는 지속적으로 이루어져야 하겠다. 또한 한정된 범위 내에서 시뮬레이션을 실시한 본 연구의 모델에 HLA 기반으로 추가적인 페더레이트와 페더레이션 개발로 안벽에서부터 게이트에 이르기까지 종합적인 컨테이너 터미널 시뮬레이션 모델 구축에 관한 연구도 지속되어야 할 것이다.

참고문헌

김창곤, 윤동환, 최종희, 배종욱, 양창호 (2000), *시뮬레이션 모델을 이용한 컨테이너 터미널 안벽능력 분석*, 한국해양수산개발원.

이상헌 (2000), HLA 모의구조 전환에 따른 한국군 DM&S 발전방향, *한*

국국방경영분석학회지, **26**(2), 101-118.
이상헌, 이영구 (2002), 상위체계구조에 근거한 수송이동관리 시제모형, *한국시뮬레이션학회논문지*, **11**(2), 31-43.
장성용, 박진우 (1988), 시뮬레이션 기법을 이용한 컨테이너 터미널의 운영시스템 결정, *산업공학지*, **1**(1), 49-64.
Atsuo Ozaki, Masakazu Furuichi, Nobuo Nishi, Etsuji Kuroda (2000), The Use of High Level Architecture in Car Traffic Simulation, *IEICE TRANS. INF. & SYST.*, E83-D(10), 1851-1859.
Department of Defense (1998), *High Level Architecture RTIAmbassador Reference RTI 1.3*, USA.
Department of Defense (1998), *High Level Architecture Supporting Class Reference RTI 1.3*, USA.
Department of Defense.(1998), *High Level Architecture Federation Development and Execution Process (FEDEP) Model Version 1.4*, USA.
Department of Defense (1998), *High Level Architecture Object Model Development Tool(OMDT) User's Guide Version 1.3*, USA.
Frederick Kuhl, Richard Weatherly, and Judith Dahmann (2000), *Creating Simulation Systems : An Introduction to the High Level Architecture*, Prentice Hall PTR.
Gambardella, L. M., Rizzoli, A. E., and Zaffalon, M. (1998), Simulation and Planning of Intermodal Container Terminal, *Simulation*, **71**(2).
Hayuth, Y., Pollatscheck, M.A., Roll, Y. (1994), Building a Port Simulator, *Simulation*, **63**(3), 179-189.
Judith S. Dahmann, Frederick Kuhl, Richard Weatherly (1998), Standards for Simulation: As Simple as Possible but not Simpler the High Level Architecture for Simulation, *Simulation,* **71**(6), 378-387.
Nevins, M. R., Macal, C. M., and Joines, J. C. (1998), A Discrete-Event Simulation Model for Seaport Operation, *Simulation*, **70**(4).

이 상 헌
육군사관학교 전자공학 학사
US Naval Postgraduate School Operations Research 석사
Georgia Institute of Technology 산업공학 박사
현재: 국방대학교 운영분석학과 교수
관심분야: 시뮬레이션, 네트워크, 메타휴리스틱, 로지스틱스

이 찬 우
육군사관학교 토목공학 학사
국방대학교 운영분석(OR) 석사
현재: 보병제61사단 공병중대장
관심분야: 시뮬레이션, 아키텍처