한국시뮬레이션학회 논문지
Vol. 17, No. 1, pp. 33-42 (2008. 3)

계층적 애니메이션이 가능한 분산 시뮬레이션 모델링 환경

이미라[1†] · 김형종[2†]

Modeling Environment for Distributed Simulation with Hierarchical Animation

Mi-Ra Yi · Hyung-Jong Kim

ABSTRACT

In general, simulation is to predict or evaluate some systems that are hard to be executed in real world, and so usually the target systems to be modeled are large and complex. Trying to observe the dynamics of the systems results in similar level of animation complexity, the model and animation has the same complexity as the system. Trying to display all the graphic objects representing the dynamics of the models being simulated, however, causes the distraction of focus, which results in solving the above listed problems difficult. The redundant graphic objects also increase the computer computation overhead. To solve the problem, a research about a hierarchical animation environment has been proposed a few years ago. In the research, the users can have better focus on the dynamics of system components by selectively choosing the hierarchical level and components within a level of the hierarchically structured model. However, the research has not a modeling methodology for modelers to describe systematically animation part corresponding to dynamics of simulation in a model. This research has defined the modeling methodology of DESHA and defined DESHA-C++, improving the previous research output, as an execution environment of DESHA models. In addition, to use hierarchical animation environment in various problems, this research proposed and developed the distributed simulation modeling environment that connects DESHA environment and HLA.

Key words : Hierarchical animation, Distributed simulation, DEVS, DESHA, HLA

요 약

시뮬레이션이 현실적으로 수행하기 어려운 시스템을 모의 실험한다는 특성상 대상 시스템은 크고 복잡한 경우가 많으며, 이러한 시스템의 변화를 관찰하고자 한다면 시뮬레이션 모델구조와 애니메이션 또한 그와 비슷한 복잡도로 행해져야 한다. 그러나, 모든 시스템 구성 요소들을 한꺼번에 애니메이션 하는 일은 그 복잡도 때문에 관찰하고자 하는 시스템 영역에 대한 초점을 흐리게 할 수 있다. 또, 시뮬레이션이 많은 컴퓨팅 자원을 요구하는 일인 것을 고려할 때, 복잡한 애니메이션 처리는 시스템에게 더욱더 많은 부하를 준다. 이와 관련한 문제를 해결하고자 전체 시스템 중 관찰하고자하는 특정 계층을 사용자가 선택하여 애니메이션 하는 계층적 애니메이션 기법이 제안되었다. 하지만, 이 연구에는 모델 작성자가 시뮬레이션의 동적 특성에 맞는 애니메이션 모델링을 체계적으로 기술 할 수 있는 모델링 형식은 정의되어 있지 않다. 이 연구에서는 계층적 애니메이션이 가능한 시뮬레이션 모델을 좀 더 체계적으로 정의 할 수 있도록 DESHA 형식론을 정의하고, 기존 연구의 구현물을 개선하여 DESHA 모델 실행 환경인 DESHA-C++로 정의하였다. 또, 이러한 계층적 애니메이션이 가능한 시뮬레이션 모델링 방식이 다양한 영역에서 활용될 수 있도록 DESHA와 HLA를 연동한 분산 시뮬레이션 모델링 환경을 제안하고 개발하였다.

주요어 : 계층적 애니메이션, 분산 시뮬레이션, DEVS, DESHA, HLA

* 이 논문은 2006년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임 (KRF-2006-331-D00446)

2008년 3월 2일 접수, 2008년 3월 15일 채택

[1] 목포해양대학교 해양전자통신공학부 조교수
[2] 서울여자대학교 컴퓨터학부 전임강사
주 저 자 : 이미라
교신저자 : 이미라, 김형종
E-mail; yimira@mmu.ac.kr, hkim@swu.ac.kr

1. 서 론

신뢰성 있는 시뮬레이션을 실현하기 위해서는 개발자가 수행하는 모델코드에 대한 검증, 개발자와 시스템 전문가(시뮬레이션 요구자) 사이의 모델 검증, 시뮬레이션 관찰자 입장에서 시스템의 동적인 변화를 이해하는 정도, 개별적인 시뮬레이션의 훈련 등이 논제로 다뤄진다[1,2]. 이와 관련한 문제들을 개선하기 위해 시뮬레이션 툴 개발자들은 시뮬레이션 과정을 애니메이션으로 출력하는 것에 관심을 갖는다[3]. 컴퓨터 애니메이션은 대상 시스템의 동적인 특징(dynamics)을 움직이는 그래픽 객체로 표현하는 출력의 한 형태로서, 대상 시스템을 직관적으로 인식하여 쉽게 이해하도록 한다[4]. 시뮬레이션 이해의 용이성은 시뮬레이션에 익숙하지 않은 시스템 전문가가 시뮬레이션 모델링 과정에 참여 할 수 있도록 유도하여 모델에 대한 신뢰도를 높이는 역할을 할 뿐 아니라, 시뮬레이션 개발자 측에서도 시뮬레이션 진행과정을 직관적으로 이해함으로써 모델에 대한 검증을 좀 더 쉽게 하도록 한다[1,5]. 실제로 상용화된 많은 시뮬레이션 개발 환경들이 애니메이션 환경을 제공하고 있으며 이러한 소프트웨어에는 Arena, MODSIM III, SIMAN, AutoMod, Factor-AIM, ProModel 등이 있다[6,12].

한편, 단일 호스트에서 실행하기 어려운 시뮬레이션을 다중 호스트로 처리 하는 분산 시뮬레이션은 병렬 처리를 통해 대규모(large-scaled) 시뮬레이션의 성능 향상 및 분산되어 있는 정보의 효과적인 이용을 가능하게 한다. 이러한 분산 시뮬레이션은 70년대 분산 처리기 및 병렬 알고리즘이 개발되면서 연구되기 시작하였고, 80년대 국방분야에서 전략 시뮬레이션 및 군사 훈련용으로 개발된 응용 프로그램인 SIMNET을 시작으로 실질적인 활용이 이루어졌으며, 이후 여러 연구 및 상용화된 시뮬레이션 도구까지 소개되어 오고 있다[7]. 분산 시뮬레이션에서는 동일한 목적의 시뮬레이션에 참여하는 서로 다른 호스트들에서 실행되는 서브모듈간의 시간(time) 개념을 반영한 상호운용성(interoperability)을 보장해주는 것을 핵심 기술로 요구하는데, 이를 위해 미국방성에서 개발한 HLA (High Level Architecture)가 현재 분산 시뮬레이션의 국제표준으로 되어있다. 또, HLA는 이기종간 가상 네트워크 세계 구성 및 실시간 실행을 가능하게 하기 때문에 국방분야 외의 일반 IT분야에서도 HLA를 유용한 기술로 활용하기 위한 연구 및 개발이 활발하게 이루어지고 있다[8-10].

시뮬레이션이 현실적으로 수행하기 어려운 시스템을 모의 실험한다는 특성상 대상 시스템은 크고 복잡한 경우가 많으며, 이러한 시스템의 변화를 관찰하고자 한다면 시뮬레이션 모델구조와 애니메이션 또한 그와 비슷한 복잡도로 행해져야 한다. 그러나, 모든 시스템 구성 요소들을 한꺼번에 애니메이션 하는 일은 그 복잡도 때문에 관찰하고자 하는 시스템 영역에 대한 초점을 흐리게 할 수 있다. 또, 시뮬레이션이 많은 컴퓨팅 자원을 요구하는 일인 것을 고려할 때, 복잡한 애니메이션 처리는 시스템에게 더욱더 많은 부하를 준다. 선행 연구인 [11]에서는 이러한 문제들을 해결하고자 전체 시스템에서 관찰하고자 하는 특정 계층을 사용자가 선택하여 애니메이션 하는 계층적 애니메이션 기법을 제안하였다. 그러나, 모델 작성자가 시뮬레이션의 동적 특성에 맞는 애니메이션 모델링을 체계적으로(명확하면서 쉽게) 기술 할 수 있는 모델링 형식은 정의되어 있지 않다.

분산 시뮬레이션이 필요한 경우는 대부분 규모가 큰 시뮬레이션이기 때문에 시뮬레이션 진행 과정을 효과적으로 보이기 위해서는 애니메이션 기능을 활용한 시각화가 더욱 필요하며 이와 관련한 연구들이 소개되었다[13,14]. 그러나, 이러한 연구들이 복잡한 대상 시스템의 특정 계층을 선택하여 애니메이션 범위를 정할 수 있는 것은 아니다.

본 연구에서는 기존 계층적 애니메이션이 가능한 시뮬레이션 모델을 좀 더 체계적으로 정의 할 수 있도록 모델링 형식론을 정의하고, 이러한 계층적 애니메이션이 가능한 시뮬레이션 모델링 방식이 다양한 영역에서 활용될 수 있도록 HLA 기반의 분산 시뮬레이션 모델링 환경을 제안하고 개발하였다.

2. 관련 연구

이 장에서는 연구의 이론적 배경인 DEVS 형식론에 대해 간략히 기술하고, 기술적 배경이 되는 시뮬레이션의 계층적 애니메이션 표현과 HLA 기반 분산 시뮬레이션에 대해 소개한다.

2.1 DEVS 방법론

DEVS 형식론은 연속적인 시간상에서 발생되는 이산 사건을 처리하는 시스템을 표현하기 위해 이론적으로 잘 정립된 모델링 방법론이다[6]. 이는 모델의 구조와 행동을 시뮬레이션 수행으로부터 추상화시키기 위하여 집합 이론을 바탕으로 정의한 모델링 방법으로서, 시스템을 계층적(hierarchical)이고 모듈화(modular)된 형태로 표현하기 용이하게 해준다. 이러한 특징을 DEVS에서는 두 가지

모델 유형- 기본(Basic)모델과 결합(Coupled)모델 -로 제공하는데, 기본모델(M)은 시스템의 동적인 특성을 표현하기 위한 것이고, 결합모델(DN)은 시스템의 구성 요소 간의 상호 작용을 표현하기 위한 것이다. 여기서 결합모델은 더 큰 단위의 상위 모델의 기준에서는 구성 요소의 단위로 사용이 가능하여 시스템을 계층적으로 표현 할 수 있게 한다.

다음은 기본 모델과 결합모델의 정의이다.

$M = < X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a >$

- X : 입력 이벤트의 집합
- S : 상태집합
- δ_{int} : 내부 상태변이 함수, $S \rightarrow S$
- δ_{ext} : 외부 상태변이 함수, $Q \times X \rightarrow S$
  (단, Q는 (s, e), $s \in S$, e: elapse-time)
- λ : 출력 함수
- t_a : 시간 갱신 함수

$DN = < D, \{M_i\}, \{I_i\}, \{Z_{i,j}\}, select >$

- D : 구성요소 이름
- M_i : 구성모델
- I_i : 모델i와 연관된 모델의 집합
- $Z_{i,j}$: 모델 i와 j모델간의 연결함수
- select : tie-breaking selection 함수

2.2 계층적 시뮬레이션과 애니메이션

선행 연구인 [11]에서는 시뮬레이션을 효과적으로 보이기 위한 계층적 애니메이션 환경을 제안하였다. 여기에서 계층성은 그래픽적 상세 정도나 출력 이미지의 우선순위가 아닌, 애니메이션 대상 시스템의 내부를 들여다보는 관점에 따라 특정 계층 또는 구성요소(tree형태의 시스템 구조에서)를 선택하여 애니메이션 함으로써 객체의 현재 상태를 표현하는 개념을 소개하였다. 여기서 애니메이션에 필요한 동적 정보(dynamics)는 모델에 명시되어 시뮬레이션에 의해 스케줄링 된 동적 결과에 의존한다. 그림 1은 논문에 소개됐던 예시 화면으로서, 간단한 공정 시스템의 시뮬레이션 진행 과정을 세 단계로 관찰했을 때의 화면을 보인 것이다. Level 1에서는 자재입고, 공정, 제품출고 간의 연계 상황만을 관찰하고, Level 2에서는 자재입고, 공정, 제품출고 각각의 내부 상황에 관심이 있는 경우의 출력형태이며, Level 3은 공정내부의 각 부품 공정까지 관찰하고자 할 때의 출력 형태이다.

계층적 애니메이션이 가능한 시뮬레이션 환경은 시뮬레이션 모델 자체가 체계적으로 계층화 되어야 효과적이기 때문에 DEVS 형식론을 기반으로 설계되었으며, 그림 2는 DEVS 기반 모델이 실행되는 환경에서의 시뮬레이터와 애니메이터의 계층적 구조의 한 예를 보인 것이다. 모델 구조에서 BM1, BM2, BM3는 기본 모델이고 CM1, CM2는 결합 모델이다. 모델 유형에 따라 시뮬레이션과 애니메이션을 위한 프로세서 또한 S(Simulator)와 A(Animator), C(Coordinator)와 CA(Co-Animator)의 형태로 구분되어 대응되어 실행된다. 모델의 개수와 무관한 최상위 프로세서인 R(Root-Coordinator)와 RA(Root-coAnimator)는 전체적인 시간관리를 맡는다.

계층적 애니메이션 개발 환경을 포함한 전체적인 시뮬레이션 개발 환경은 그림 3와 같이 제시되었다. 모델링을 할 때 시뮬레이션과 애니메이션에 관련된 것들은 각각 DEVS-C++ 라이브러리, 자체 개발한 Animation Class 라이브러리를 이용하여 작성하고, 애니메이션 관련 모델링을 할 때 필요한 그래픽 관련 파일들은 Graphic Editor로 생성된다. 이렇게 하여 생성된 시뮬레이션 모델이 컴파일 되면 사용자가 작성한 모델의 정보를 유지하는 Model, 이벤트 스케줄링을 하는 프로세서인 Simulator, 이벤트 스케줄링을 가시화하기 위한 프로세서인 Animator

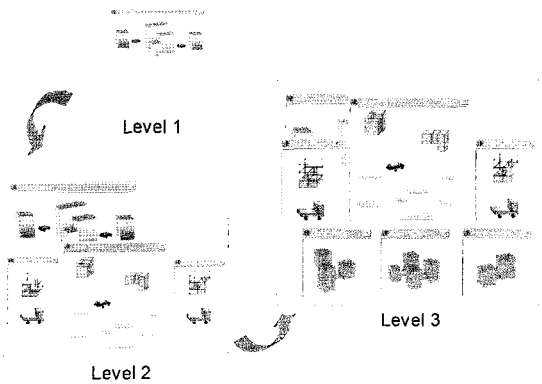

그림 1. 계층적 애니메이션의 예

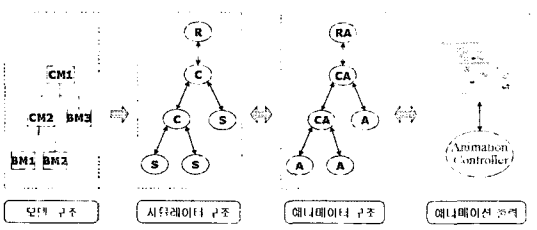

그림 2. 계층적 모델링, 시뮬레이션, 애니메이션

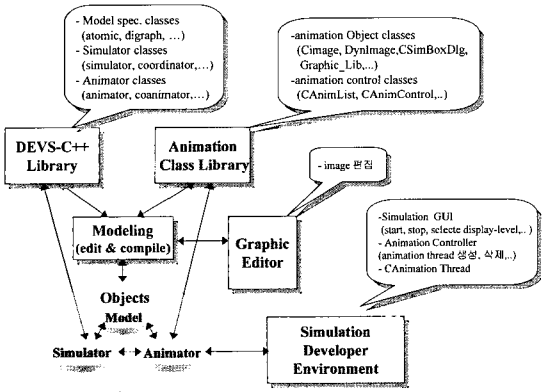

그림 3. 시뮬레이션 애니메이션 개발 환경 구성도

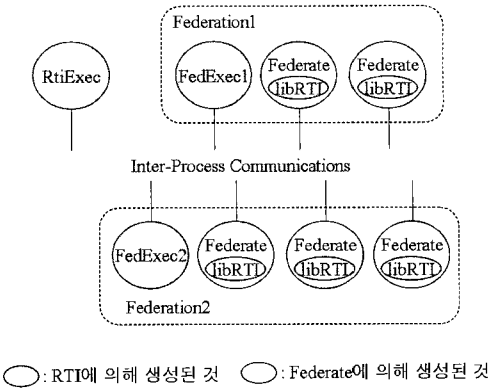

그림 4. RTI 구성요소 개념도

들이 생성되고, 실제로 시뮬레이션 진행은 이러한 프로세서들 간의 상호작용을 통해 이루어진다.

2.3 HLA 기반 분산 시뮬레이션

HLA가 미 국방부에서 연구 개발된 것인 만큼, 미 국방부에서는 현재 모든 시뮬레이션 모델에 대해 HLA 준수를 강제화하기로 결정하고, 국제 표준기구인 IEEE도 이 기술을 네트워크 가상환경 국제표준으로 채택하였다. 우리나라의 국방부도 최근 육군 전투시뮬레이션 게임을 HLA 기반으로 개발하는 등 향후 국방정보화 전 분야에서 HLA를 근간기술로 채택한다는 방침을 세웠다[15]. 미국 등 선진국에서는 HLA가 가상전쟁게임등 국방분야 뿐만 아니라 일반 정보화 분야에도 응용가능성이 큰 것으로 기대를 받음에 따라 이를 연구·개발하는 전문가들이 크게 늘고 있다. 국내에서는 지능형교통시스템(ITS)를 위한 기반 기술 개발[16], 선박의 안전성 평가 시뮬레이션 프레임워크[17], 임베디드 시스템을 위한 시뮬레이션 환경[18] 등의 연구들이 이에 해당한다.

HLA 기반 시뮬레이션의 구체적인 구성요소는 (1) Federation규칙, (2)인터페이스명세(RTI: Run-Time Infrastructure), (3)객체모델템플릿(OMT: Object Model Template) 등으로 정의된다[20]. 여기서 Federation 이란 각 호스트에서 실행되는 시뮬레이션 단위인 Federate 구성요소들이 논리적으로 결합되어 완성된 하나의 시뮬레이션 단위를 말한다. 즉, HLA내에서 Federation은 객체(object)와 상호작용(interaction) 형태로 정보를 교환하는 Federate으로 구성된다. HLA의 세 구성 요소 중 RTI(Run-Time Infrastructure) 소프트웨어는 인터페이스 명세를 구현한 것으로 HLA의 대표적인 산출물이다.

RTI는 분산 운영체제가 응용 프로그램에게 서비스를 제공하는 것과 유사한 방법으로 서비스를 제공한다. 그림 4는 RTI의 구성 요소인 RtiExec, FedExec, libRTI의 관계를 보여주고 있다. RTI 소프트웨어는 독립적인 워크스테이션이나 또는 매우 복잡한 네트워크 상에서도 실행될 수 있다. RtiExec프로세스는 Federation 실행의 생성과 파괴를 관리하여 하나의 네트워크 상에서 다수의 Federation들이 서로 다른 이름으로 실행 될 수 있도록 하고, 개별 실행 Federation은 하나의 전역적 프로세스인 FedExec에 의해 특징지어진다. FedExec는 Federate이 Federation에 가입하고 탈퇴하는 것을 관리한다. C++로 구현된 libRTI 라이브러리는 Federate 개발자에게 RTI 서비스를 제공하며, 서비스는 libRTI, RtiExec 및 적당한 FedExec 간 캡슐화 통신을 통해 이루어진다.

3. 계층적 애니메이션이 가능한 분산 시뮬레이션 환경

3.1 전체 개념

이 연구의 시뮬레이션 모델링 방법은 DEVS 형식론을 확장하여 DESHA(Discrete Event system Specification with Hierarchical Animation)를 정의하여 이용한다.

연구 내용은 크게 두 단계로 나눌 수 있다. 첫 번째 단계는 DESHA를 정의하고 DESHA 모델 실행 환경인 DESHA-C++를 개발하는 것이다. 계층적 애니메이션을 시뮬레이션 모델 정의 시 용이하게 명시할 수 있도록 DEVS를 확장하여 DESHA 형식론을 정의한다. 이렇게 정의된 DESHA를 기존의 계층적 애니메이션 환경에서 반영 될 수 있도록 프로그램을 개선하여 검증 할 수 있는

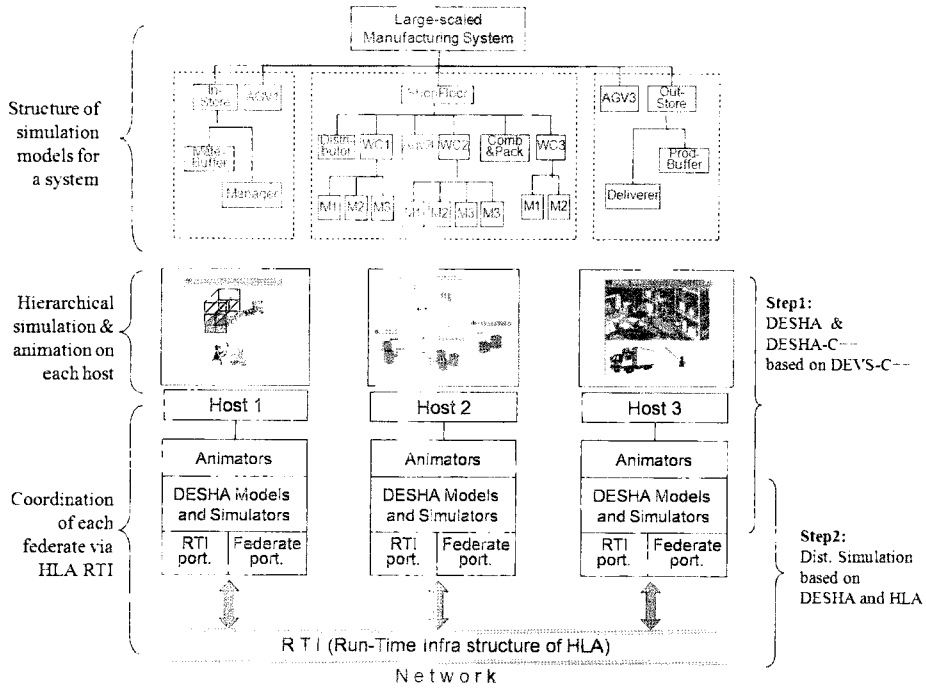

그림 5. 계층적 애니메이션과 분산 시뮬레이션의 연동

도구인 DESHA-C++을 개발한다. 두 번째 단계는 DESHA 기반의 분산 시뮬레이션 환경을 개발하는 것으로서, DESHA -C++가 분산 시뮬레이션을 위한 모델로 활용될 수 있는 환경을 구축하는 것이다. 이를 위해 분산 시뮬레이션을 위한 하부 구조로 사용하기 위한 DESHA-C++와 HLA가 연동하여 분산 시뮬레이션을 실행 할 수 있는 환경을 설계 및 구현한다. 그림 5[1]는 이 두 단계의 내용을 그림 1의 대상 시스템을 예로 표현한 개념도이다.

DESHA 모델의 실행 환경인 DESHA-C++은 선행 연구 [11]의 시뮬레이션 환경을 개선하는 방식으로 정의하여 개발하였다. 애니메이터와 연동되는 주요 알고리즘은 그대로 유지하고, 기존 DEVS모델과 DESHA 모델을 구분하여 클래스를 추가 및 수정하였다. 또, 함수와 변수들의 이름 등을 적절하게 변경하였고, 애니메이션 객체 관련 라이브러리와 애니메이터 관련 라이브러리를 통합하여 모델링 환경을 단순화 하였다.

분산 시뮬레이션 환경을 위해서는, HLA(High Level Architecture)에서 제공하는 RTI 1.3ng v6[19] 스펙에 맞추어 DESHA-C++를 추가 확장한다.

1) 그림의 RTI port 와 Federate port에서 'port'는 HLA/RTI에서 'Ambassador'로 표기된다.

3.2 DESHA 정의 및 모델링 환경

3.2.1 DESHA 정의

앞서 2.1에서 이론적 접근방법으로 소개한 DEVS 형식론을 확장하여 애니메이션 출력이 가능한 모델을 정의하기 위하여, 기본모델에 해당하는 M을 M_A으로 확장하고, 결합모델에 해당하는 DN을 DN_A으로 확장하였다. 확장하여 정의한 M_A와 DN_A는 다음과 같다.

$M_A = \langle X, S_A, Y, \delta^A_{int}, \delta^A_{ext}, \lambda, t_a, DIs, SIs, Wnd \rangle$

이 때,

- X : 입력 이벤트의 집합
- S_A : $S \times A$ (단, S는 상태 집합, A는 애니메이션 동작 상태 집합)
- Y : 출력 이벤트의 집합
- δ^A_{int} : 내부 상태변이 함수, $S_A \dashrightarrow S_A$
- δ^A_{ext} : 외부 상태변이 함수, $Q_A \times X \rightarrow S_A$ (단, Q_A는 (s_a, e), $s_a \in S_A$, e:elapse-time)
- λ : 출력 함수
- t_a : 시간 갱신 함수
- DIs : 동적(애니메이션) 이미지 객체 집합
- SIs : 정적(배경) 이미지 객체 집합
- Wnd : DIs와 SIs가 출력될 윈도우 화면

DN_A = < D, {M_i}, {I_i}, {$Z_{i,j}$}, select, SIs, Wnd >
이 때,
- D : 구성요소 이름
- M_i : 구성모델
- I_i : 모델 i와 연관된 모델의 집합
- $Z_{i,j}$: 모델 i와 j모델간의 연결함수
- select : tie-breaking selection 함수
- SIs : 정적(배경) 이미지 객체 집합
- Wnd : SIs가 출력될 윈도우 화면

위의 두 모델 모두 Wnd가 null이면 계층적 모델 구조에서 상위 모델의 Wnd를 기본(default) Wnd 값으로 사용하여 DIs와 SIs를 출력하게 된다.

상태변이 함수인 δ^A_{int}와 δ^A_{ext}에서는 각 모델의 상태변이 내용에 대응되는 애니메이션 동작을 지시하는 내용이 추가되어 있으며 그림 6은 이러한 상태 변이와 애니메이션 동작 매핑 관계를 보여주고 있다.

3.2.2 DESHA-C++ 개발

Devs-C++을 기초로 DESHA를 반영하여 작성한 DESHA-C++에는 그림 7에서 보이는 것과 같은 클래스들을 추가(색이 칠해진 부분)하였다.

2장에서 그림 2를 통해 설명한 계층적 모델 구조, 시뮬레이터 구조, 애니메이터 구조의 개념도를 기준으로 보았을 때, Devs-C++은 모델 구조의 개별 모델과 그에 대응되는 시뮬레이터가 하나의 클래스 안에서 함께 정의되어 있는 형태이다. 여기서, 계층적 애니메이션 관련한 내용을 사용자가 모델에 추가해야 할 속성과 시뮬레이터가 애니메이터와 연결하기위해 필요한 속성들을 추가하여 devsA, atomicA, digraphA 등을 정의하였다. 한편, 그림 2에서 애니메이터 구조에 해당하는 것들은 별도의 클래스 계층으로 정의를 하였으며, 이들이 animbase, Animator, Co-Animator, Root-coAnimator이다. 이 중 animbase, Animator, CoAnimator는 각각 devsA, atomicA, digraphA에 대응되어 실행될 애니메이션 엔진(애니메이터)에 해당되며, Root-coAnimator는 계층적 애니메이션 엔진 구조의 최상위 애니메이터에 연결되어 시뮬레이터의 이벤트 스케줄링과 연결시키기 위한 프로세서이다. 시뮬레이션 실행 시 모델 구조와 동일한 구조의 애니메이터들이 실행되고, 이들은 논리적인 연결을 통해 하나의 abstract-animator로서 동작한다. 이러한 동작 원리는 선행연구인 [11]에 상세히 기술되어 있다.

devsA에서는 DESHA 모델을 실행하기 위해 추가되어야 할 속성(멤버변수) 및 메서드(멤버함수)를 정의해 놓고 있는데, 그 주요 내용은 다음과 같다.

```
static bool        m_bIsRoot;
CObList *          imgObjs; // image objects
CAnimList *        animObjs; //dynamic image objects
CAnimControl*      animCtrl;
bool               trsFlag;
bool               m_bDisp; // Display or Hidden?
bool               m_bDoneAnim;
CWnd *             m_pDispWnd;
animbase *         m_pCorrAnimator;
rootcoanimator*    m_pRootcoanimator;
virtual void SetDispWnd(CWnd *);
virtual void A_info(CDynImage*, CPoint, int);
virtual CAnimList * get_A_info();
virtual void I_info(CImage * iObj);
virtual CObList * get_I_info();
virtual void SetDispMode(bool disp);
```

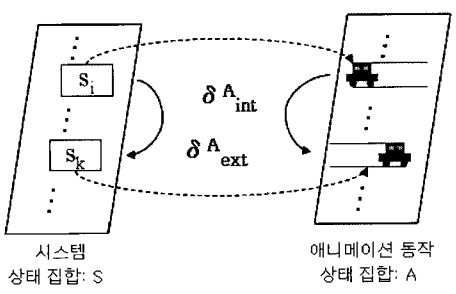

그림 6. 상태 변이와 애니메이션 동작 매핑 관계

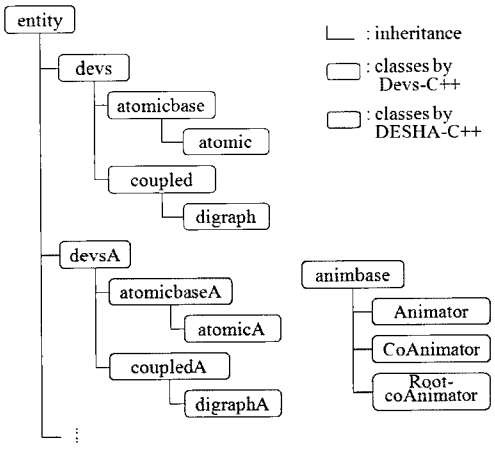

그림 7. DESHA-C++의 클래스 계층도

```
virtual bool GetDispMode();
virtual void DisplayImages();
virtual void CompDisplayImages();
virtual void ReceDoneAnim();
virtual void RunSimulCycle();
```

devsA에서는 추가된 메서드들이 모두 가상 함수 형태이며, atomicA에서는 이들 모두를 구현하고, coupledA에서는 동적이미지 관련 함수들(A_info, get_A_info)을 제외한 모든 함수를 구현해 놓았다. 각 모델의 시뮬레이터 역할을 하는 기존 devs 클래스의 메소드들 중 일부도 애니메이터와의 연결을 위해 수정을 하였는데, 이는 본 연구와 관련된 이전 연구 결과물인 [11]에 기술되어 있다.

3.3 DESHA 기반의 분산 시뮬레이션 환경 개발

시뮬레이션의 계층적 애니메이션이 분산 환경에서도 가능하도록 DESHA-C++와 HLA가 연동되는 분산 시뮬레이션 환경을 설계 및 구현하였다.

HLA에서는 RTI 소프트웨어를 통해 네트워크 상에 있는 단위 시뮬레이션(federate) 간 인터페이스를 가능하게 하므로, DESHA 환경이 HLA와 연동하기 위해서는 이 RTI 소프트웨어와의 연결 관계를 생각해보아야 한다. DESHA 환경에서는 일반적으로 여러 개의 작은 단위 모델들을 계층적으로 결합시켜 완성된 하나의 모델 구조를 만드는데, 이 구조의 최상위 모델을 RTI와 연결시키는 모델로 작성함으로써 HLA와 연동 할 수 있다. 그림 8는 이러한 개념을 표현한 것이며, 여기에서 각 federate에 있는 모델 이름 및 모델 내의 구성요소 이름은 그림 9처럼

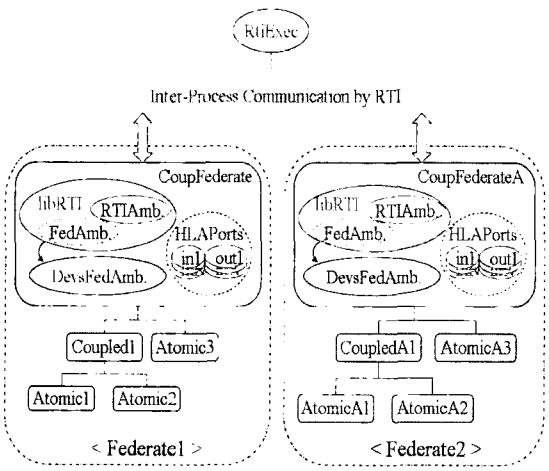

그림 8. DESHA-C++와 HLA의 연동

DESHA-C++을 확장한 클래스들의 동일 이름에 해당하는 인스턴스를 의미한 것이다. 그림 8에서 왼쪽의 CoupFederate은 애니메이션 출력이 없는 기존 Devs 모델들로 구성된 federate의 예를 보인 것이고, 오른쪽의 CoupFederateA는 애니메이션 출력이 있는 DESHA 모델들로 구성된 federate의 예를 보인 것이다. 그림에서는 최상위 모델이 결합모델로만 되어있으나, 기본모델 하나만으로 federate이 될 수도 있다.

CoupFederate, CoupFederateA, Federate, FederateA는 libRTI의 RTIAmbassador를 통해 RTI에 접속/해제 서비스를 이용하고, libRTI를 확장하여 정의한 DevsFedAmbassador를 통해 콜백함수를 제공한다. 콜백함수를 제공하는 DevsFedAmbassador클래스는 내부적으로 DESHA 모델과 연결되어 HLA와 연동되게 하는 핵심 역할을 한다. 한편, DESHA 모델간 입/출력을 위한 자료형인 port를 상속하여 RTI를 통한 federate간 입/출력을 할 수 있도록 HLAPort 자료형을 정의하여 제공한다.

4. 샘플 시스템을 통한 검증

4.1 샘플 시스템 구현

앞서 정의한 DESHA 및 DESHA-C++와 HLA 기반 분산 시뮬레이션 환경에 대한 검증을 위해 단순화 시킨 선박 건조 공정 시뮬레이션을 실행시켜 보았다.

보통의 선박 건조는 일반 건축물보다 규모도 크고, 수

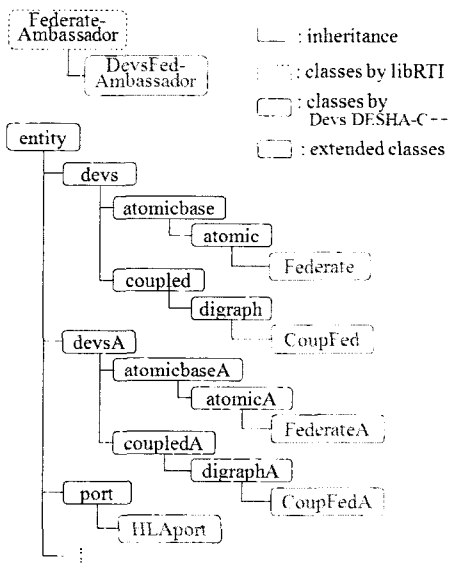

그림 9. HLA와의 연동을 위한 DESHA-C++ 확장

많은 부재와 기자재를 조립하는 과정으로 인하여 훨씬 복잡한 공정 과정을 거치게 된다. 그림 10은 선주와 조선소 간의 견적을 통한 계약부터 건조 완성 후 인도될 때까지의 선박 건조 과정을 간단하게 그린 것이다[21]. 일반적으로 조선소에서는 여러 선박 건조를 동시에 진행하며, 하나의 선박을 건조하기 위한 세부 공정들이 병렬로 진행되기도 한다. 또, 부재 및 기자재를 여러 협력 업체를 통해 조달하기도 한다.

선박 건조 공정은 수많은 인력, 프로세스, 데이터 및 자재가 최적의 조합을 이루어야 하는 복합 산업이기 때문에 설계부터 생산까지의 여러 공정을 효율적이고 유연하게 통합하여 관리할 수 있는 환경을 마련하는 것이 중요하며, 이를 위해서는 다양한 소프트웨어 기술들이 뒷받침되어야 한다[22]. 특히, 복잡한 공정에 따른 결과 예측 및 평가를 위한 컴퓨터 시뮬레이션 기술의 효용성도 널리 인식되어 있으며 이와 관련한 연구도 꾸준히 진행되어오고 있다[23,24] 또, 향후의 설계, 구매, 생산 환경이 전 세계적으로 분산된 환경에서 가상환경을 통해 협업해야 하는 형태를 고려하여 분산 시뮬레이션 기술 표준(HLA)등에 대한 적용 준비

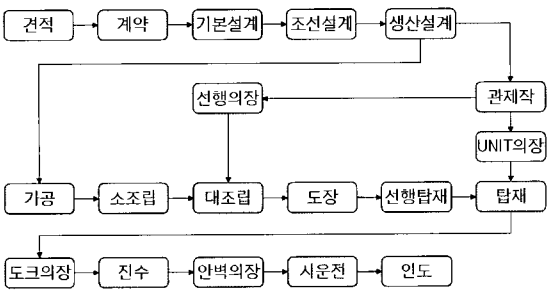

그림 10. 선박 건조 공정 개요

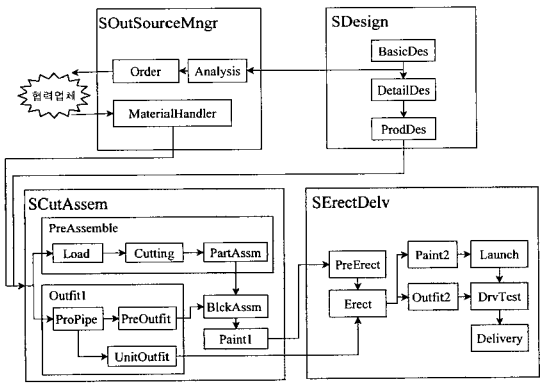

그림 11. 선박 건조 공정 시뮬레이션 모델 구조도

도 필요하다[22].

본 연구에서는 그림 10을 기초로 하여 그림 11과 같은 간단한 시뮬레이션 모델 구조를 설계하였다. 그림에서 보이는 것과 같이 전체 모델 구조는 네 개의 모델 - SDesign (설계과정), SOutSourceMngr(협력업체를 통한 자재 및 부품 조달과정), SCutAssem(절단 및 조립 과정), SErectDelv (탑재 및 진수 과정) - 들로 결합되어 있는데, 이 네 모델이 독립적인 federate 단위로 실행될 모델이다. 이 중 SDesign와 SOutSourceMngr는 애니메이션 출력이 없는 federate으로서 CoupFederate 클래스의 인스턴스 형태의 모델이고, SCutAssem와 SErectDelv는 애니메이션 출력이 있는 federate으로서 CoupFederateA 클래스의 인스턴스 형태의 모델이다. 또, SCutAssem은 3계층으로 구성되었고, SErectDelv는 2계층으로 구성되었다. 이러한 모델 구조의 분산 시뮬레이션을 실행하였을 때의 화면 예시는 그림 12와 같다. 네트웍으로 연결된 네 개의 컴퓨터에 각각 federate 하나씩 실행되고, 이중 SCutAssem이 두 단계의 계층적 애니메이션 출력을 보이고 있다.

4.2 결과 고찰

이 연구의 이론적 결과물은 계층적 애니메이션이 가능한 시뮬레이션 모델링 방법을 형식화하여 정의한 DESHA 형식론이고, 구현 결과물은 DESHA 모델을 실행시킬 수

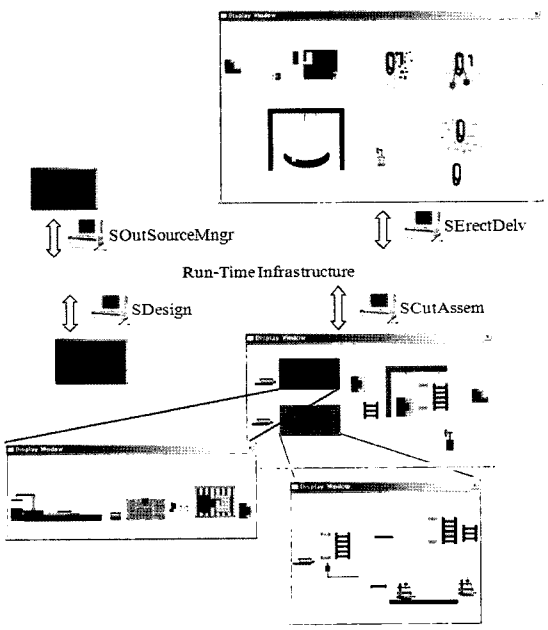

그림 12. 선박 건조 공정 시뮬레이션 화면 예

있는 환경인 DESHA-C++을 정의하고, DESHA모델이 다양한 영역에서 활용될 수 있도록 하기 위해 DESHA-C++환경을 HLA 기반의 분산 시뮬레이션 모델링 환경으로 확장한 것이다. 샘플 모델 구현을 통해 DESHA가 모델 작성자에게 계층적 애니메이션에 대한 명확한 개념적 지침이 제공 할 수 있음을 확인 할 수 있었고, HLA를 연동할 수 있도록 확장한 DESHA-C++ 환경이 설계 내용에 적합하게 구현되어 실행됨을 확인 할 수 있었다. 그러나, DESHA처럼 분산 시뮬레이션 환경을 반영하여 이론적인 확장까지 하지는 않았기 때문에, 모델 작성자는 기존 HLA 관련 지침서를 함께 참고해야 하는 환경이 되었다. 향후, 모델 작성자가 분산 시뮬레이션 환경과 연동되는 최상위 모델의 정의를 좀 더 쉽게 할 수 있도록 개선이 필요할 것으로 보인다.

5. 결 론

이 연구는 시뮬레이션 진행 과정을 효과적으로 표현하기 위한 계층적 애니메이션 기법을 제안했던 기존 연구의 후속 연구이다. 기존 연구 기법을 모델 작성자가 시뮬레이션의 동적 특성에 맞는 애니메이션 모델링을 체계적으로 정리하여 DESHA로 정의하고, 계층적 애니메이션 라이브러리를 이용하여 DESHA 모델 실행 환경인 DESHA-C++로 수정하였다. 또, 이러한 계층적 애니메이션이 가능한 시뮬레이션 모델링 방식이 다양한 영역에서 활용될 수 있도록 DESHA와 분산 시뮬레이션 표준 기술인 HLA를 연동한 분산 시뮬레이션 모델링 환경을 제안하고 개발하였다. 분산 환경에서의 DESHA 모델링 및 실행 환경은 선박 제조 공정을 단순화 시킨 샘플 모델을 통해 검증해 보았다. 연구 결과는 시뮬레이션 모델 작성자에게는 모델 실행을 시각화 시키는데 필요한 요소들을 명확히 안내해 줄 수 있을 것으로 기대되고, 시간 개념이 반영된 여러 시각화 분야에서의 시스템 모델링 기술의 접근 방법으로 활용 될 수 있을 것으로 기대된다. 한편, DESHA 모델 실행 환경으로 개발된 DESHA-C++ 결과물은 모델링 방법을 확인해볼 수 있는 환경으로 활용 할 수는 있는 수준인데, 차후에는 사용자 인터페이스를 개선하여 실제 상용화 가능한 도구로까지 활용 될 수 있도록 개선이 필요하다.

참 고 문 헌

1. David R. Hill, "Object-Oriented Analysis and Simulation", Addison-Wesley, 1996.
2. Averill M. Law and W. David Kelton, "Simulaiton Modeling and Analysis, Second Edition", McGraw-Hill, 1991.
3. Siroos Sokhan-Sanj ,Gerald T. Mackulak, "The value of Simulation Animation: Discussion of Instances Where Statistical output is Insufficient for Analysis of System Performance", Proc. of the 2nd Annula International Conference on Industrial Engineering Application and Practice II, Vol. 2, 1997, Califonia USA.
4. Nadia M. Thalmann, Daniel Thalmann, "Computer Animation", ACM Computing Surveys, Vol. 28, No. 1, March 1996.
5. Lan Sommerville, "Software Engineering", Fifth Edition, Addison -Wesley, 1995.
6. Bernard P. Zeigler, "Object-Oriented Simulation with Hierarchical, Modular Models", Academic Press, 1990.
7. 정재훈, "A Survey on the Distributed Interactive Simulation(DIS) and the High Level Architecture(HLA)", Technical Note, http://vr.kaist.ac.kr, 1998.
8. 장윤옥, "군 시뮬레이션 2개 모델 국제표준 연동체계 인증", 디지털타임즈, 2003.
9. Kim, Y. J., Cho, J. H. and Kim, T. G., "DEVS-HLA: Heterogeneous Simulation Framework Using DEVS BUS Implemented on RTI", Summer computer simulation conference, 1999.
10. 차영필, 정무영, "분산 생산 시스템을 위한 에이전트 기반의 협업 시뮬레이션 체계", 한국경영과학회 학술대회지, 2003.
11. Yi, M. R. and Cho, T. H., "Hierarchical simulation model with animation", Engineering with computers, Vol. 19, No. 2/3, 2003.
12. T. J. Schriber and D. T. Brunner, "Inside discrete-event simulation software: how it works and why it matters", Proc. of Winter Simulation Conference, Dec. 2005.
13. S. Singhal, "Effective Remote Modeling in Large-Scale Distributed Simulation and Visualization Environments", PhD thesis, Stanford Univ., Aug. 1996.
14. S Straburger, T Schulze, U Klein "Internet-based simulation using off-the-shelf simulation tools and HLA", Proc. of WSC98, 1998.
15. 안경애, "HLA, 응용분야 넓어 차세대 기술로 이용". 디지털타임지, 2002.
16. Lee, J.-K.; Lee, M.-w.; Chi, S.-D. "DEVS/HLA-Based Modeling and Simulation for Intelligent Transportation Systems", Simulation, Vol. 79, No. 8, 2003.
17. 이경호, 김화섭, 한선우, 박종현, 오준, "선박의 안전성 평가를 위한 네트워크 기반의 시뮬레이션 시스템 프레임워크", 한국CAD/CAM학회논문집, Vol. 10, No. 5, 2003.

18. 정재경, 김호정, 원강연, 김종룡, "내장형 시스템을 위한 HLA기반 분산 실시간 시뮬레이션 환경 구현", 한국정보과학회 03 춘계 학술발표논문집, pp. 400-402, 2003.
19. DMSO,http://www.dmso.mil/public/transition/hla
20. DMSO, RTI 프로그램 지침서.
21. 임문규, "조선 설계 및 생산기술의 발전 동향", 대한조선학회지, 39권 4호, 2002.
22. 이성근, 서흥원, 이원준, "조선산업을 변화시키는 소프트웨어 기술", 정보과학회지, 25권 2호, 2007.
23. 오대균, 이춘재, 최양렬, 신종계, 우종훈, "선박 건조 공정 시뮬레이션을 위한 모델링 방법론 및 시스템 아키텍처", 한국CAD/CAM학회논문지, 11권 1호, 2006.
24. 박주용, 김세환, 최우현, "조선소 판넬라인의 최적 생산계획 수립을 위한 생산 시뮬레이션 연구", 대한용접학회지, 24권 5호, 2006.

이 미 라 (yimira@mmu.ac.kr)

1998 성균관대학교 정보공학과 학사
2000 성균관대학교 전기전자 및 컴퓨터공학과 석사
2005 성균관대학교 전기전자 및 컴퓨터공학과 박사
2005～현재 목포해양대학교 해양전자통신공학부 조교수

관심분야 : 모델링 방법론, 인공지능과 시뮬레이션, 시뮬레이션 개발 환경, ERP, 자율운항시스템

김 형 종 (hkim@swu.ac.kr)

1996 성균관대학교 정보공학과 공학사
1998 성균관대학교 정보공학과 공학석사
2001 성균관대학교 전기전자 및 컴퓨터공학과 공학박사
2001～2007 한국정보보호진흥원 수석연구원
2004～2006 미국 카네기멜론대학 CyLab Visiting Scholar
2007～현재 서울여자대학교 컴퓨터학부 전임강사

관심분야 : 취약점 분석 및 모델링, 이산사건 시뮬레이션 방법론, 침입감내기술