메타프로그래밍 기법을 이용한 설정 가능한 분산객체 통신 모듈 설계

심준용○ 진정훈 김세환
LIG 넥스원(주)
jyshim@lignex1.com, jhjinb@lignex1.com, shkimc@lignex.com

A Design of A Configurable Communication Module of Remote Object Using Meta Programming

Junyong Shim○ Jeonghoon Jin, Sehhwan Kim
LIG Nex1 Co., Ltd.

요 약

분산 시뮬레이션 환경에서 모의되는 개체의 재사용성과 개체 간 상호운용성을 높이기 위해서 Modeling & Simulation 기법이 적용된 M&S Framework이 제안되었다. 제안된 프레임워크의 미들웨어 통신을 담당하는 시뮬레이션 네트워크 관리자는 계층적 아키텍처 스타일을 적용함으로써 관리성(maintainability), 재사용성(reusability), 확장성(scalability) 등의 요구사항을 지원하도록 설계되었다. 하지만 프레임워크를 구현한 다양한 모의 개체의 생성은 메시지의 인터페이스 코드에 대한 중복을 증가시키고, 인터페이스 변경 요구사항에 대해서 시뮬레이션 네트워크 관리자가 적용된 모든 모의기의 변경 또한 불가피해 진다.

본 논문에서는 인터페이스에 대한 코드의 중복을 없애고, 인터페이스의 변경에 대하여 유연성을 가질 수 있도록 메타프로그래밍 기법을 적용한 분산객체 통신 모듈을 제안한다. 또한 이 기법을 적용할 경우 구현 문제와 관련된 이슈와 장·단점을 기술한다.

1. 서 론

미래 전장의 환경이 과학화 / 정보화됨에 따라 육해·공군의 장비들은 필요한 정보들을 서로 공유하여 통신할 수 있는 구성체계간 상호운용성과 구성체계의 효용성을 높이고 비용 절감을 위한 재사용성의 제공이 핵심 기술로 대두되고 있다. 이러한 기술의 제공은 분산 시뮬레이션 환경 기반의 Modeling & Simulation 기법이 적용된 어플리케이션을 개발하는데 있어 시뮬레이션에 참여하는 객체들 간의 통신을 위한 핵심이 된다.

미 국방성(DoD)은 분산 시뮬레이션 환경의 상호운용성을 보장하고 모의 시뮬레이터를 포함한 실 시스템과의 연동을 용이하게 하며 시뮬레이션 소프트웨어의 재사용성을 촉진시키기 위한 방안으로 HLA(High Level Architecture) 아키텍처를 발표한다[1]. 아키텍처의 구조는 그림 1과 같다.

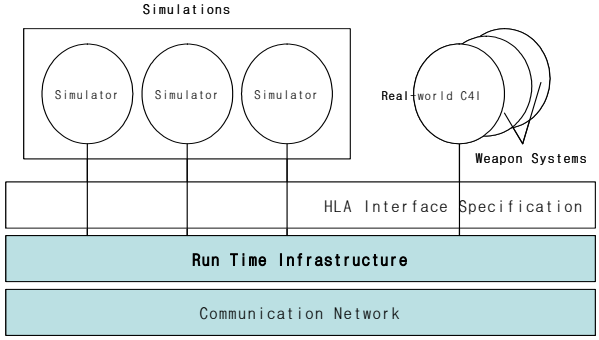

그림 1 HLA 아키텍처 구조

HLA 표준에 정의되어 있는 서비스는 RTI(Run-Time Infrastructure) 미들웨어 소프트웨어에 의해서 구현된다. RTI는 TCP/UDP/IP 프로토콜을 기반으로 분산객체 통신을 지원하며 정보의 송수신 방식은 Publish / Subscribe 메커니즘을 사용한다. RTI는 사용자에게 원하는 서비스를 사용할 수 있도록 지원하는 RTIambassador와 RTI로부터 사용자에게 제공되는 Callback 서비스를 지원받을 수 있는 FederateAmbassador 인터페이스를 제공한다. 구성요소의 구조를 추상화하면 그림2와 같다.

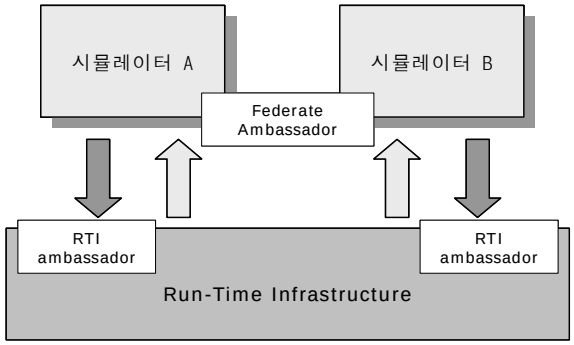

그림 2 RTI 인터페이스 구조

기존에 제안된 M&S 프레임워크의 미들웨어 통신 기능을 담당하는 시뮬레이션 네트워크 관리자는 HLA 표준을 적용하여 시뮬레이션에 참여하는 모의 개체간의 상호연동을 지원하기 위한 모듈이다. 시뮬레이션 네트워크 관리자는 모듈 간 느슨한 결합(loosely-coupled)을 유지

하는 계층적 아키텍처(Layered Architecture) 구조로 이루어져 있으며 관리성, 재사용성, 확장성 등의 요구사항을 지원하는 특징을 갖는다. 시뮬레이션 네트워크 관리자의 배치 구조는 그림 3과 같다[2].

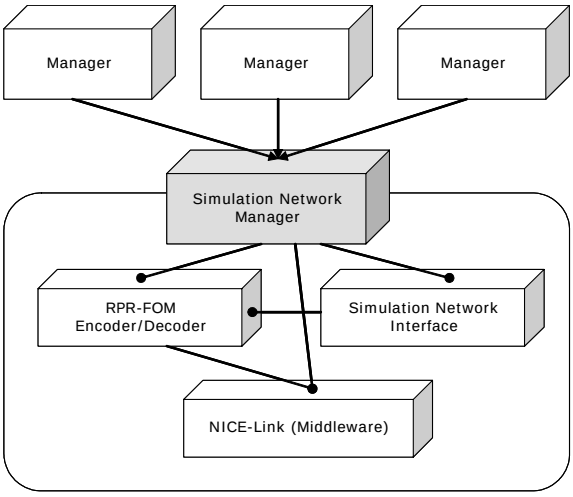

그림 3 시뮬레이션 네트워크 관리자 구조

계층적 아키텍처 스타일은 모듈에 대한 변경의 지역화(localize modification)와 관심의 분리(separation of concerns)라는 측면에서 유지보수의 용이성과 재사용성에 대한 이점을 가져다준다[3]. 하지만 모의기 간에 주고받는 메시지의 인터페이스 규약에 대한 변경사항이 발생할 경우 모든 모의기의 시뮬레이션 네트워크 관리자의 수정은 불가피해진다. 그림 4는 인터페이스의 변경에 따른 구조적인 문제를 보여준다.

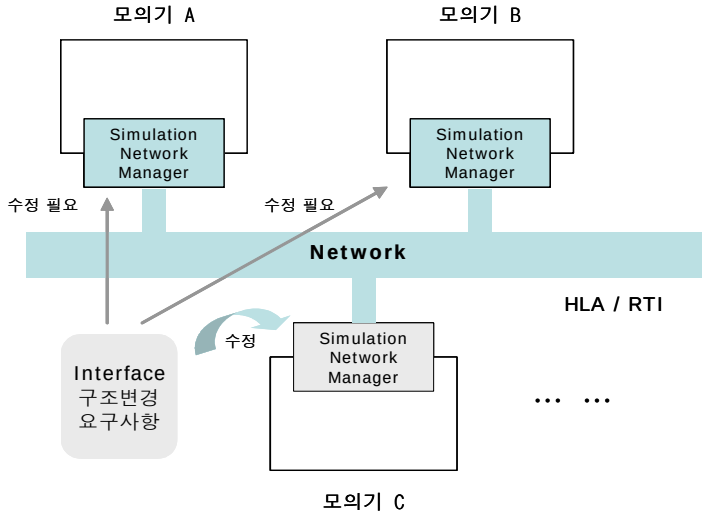

그림 4 인터페이스 변경에 따른 구조적 문제

일반적으로 인터페이스에 따른 프로그래밍은 코드 재사용의 측면에서 구현에 대한 유연성을 제공하는 반면 인터페이스 규약에 대한 변경사항은 재사용 코드의 구현 변경에 대한 문제는 해결할 수가 없다. 단독으로 수행되는 소프트웨어의 경우 미치는 영향이 크지 않지만 인터페이스를 구현한 공통모듈을 사용하는 소프트웨어가 다수인 경우 유지보수의 비용은 크게 증가하게 된다. 또한 제안되었던 분산객체 통신 모듈의 경우 실행시간(Run-Time)에 주고받는 메시지의 전송 방식의 변경에 대한 요구사항이 발생하지만 재 컴파일(Recompiling)을 해야만 하는 문제를 안고 있다.

따라서 본 논문에서는 메타 프로그래밍 기법을 적용하여 인터페이스에 대한 코드생성(Code Generating)을 통한 각 모의기 고유의 인터페이스를 생성 시켜줌으로써 인터페이스 변경 요구사항에 대한 유연성을 제공하고, 모의기 간에 주고받을 메시지의 정보가 기술된 XML(eXtensible Markup Language)형식의 문서를 통하여 실행시간에 메시지의 전송방식을 설정 가능하게 하는 분산객체 통신 모듈을 제안한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 일반적인 메타 프로그래밍의 정의와 기법들에 대해서 살펴본 후 제안하는 모듈에 대한 접근 방법을 제시한다. 제 3장에서는 본 논문에서 제안하는 분산객체 통신모듈의 아키텍처와 구성 요소의 특징 및 구현 기술을 살펴보고, 제 4장에서는 결론 및 향후 연구방향을 제시한다.

2. 관련 연구

메타 프로그래밍은 일반적으로 코드 생성(Code Generating) 프로그램이라고도 부르며, 프로그램을 위한 프로그램을 작성하기 위한 방법으로 어플리케이션의 코드를 추상화 하고 어플리케이션이 실제 수행해야 될 세부사항에 대해서는 설정 가능하도록 한다[4]. 본 장에서는 메타 프로그래밍의 기법들을 소개하고 그 특징들을 살펴본다. 또한 소개된 기법들을 통하여 제안하는 모듈의 설계 방안을 기술한다.

2.1 메타 프로그래밍 방법

2.1.1 Pre-generating tables

실행시간에 사용할 데이터를 위한 테이블을 미리 생성시킬 수 있는 프로그램을 작성하는 방법이다. 예를 들어 어떤 값 n이 있고 이 값이 컴파일 시간에 전달된다고 하면, n 차원 벡터 클래스를 만들 수 있다. 일반적인 방법으로 n 차원 벡터를 만드는 것보다 반복문이 필요 없고 상당히 최적화된 코드를 얻을 수 있는 이점이 있다.

2.1.2 Mini-language

많은 양의 반복적인 코드(boilerplate code)가 포함된 어플리케이션의 경우 반복 코드를 대신해서 프로그램 해줄 수 있는 언어(language)를 생성하는 방법이다. 컴파일하기 전에 정상적인 소스 코드(regular source code)로 변환시켜주는 매크로 프로세서(macro processor) 등이 여기에 속한다.

2.1.3 Code-generating

코드를 생성하는 코드를 작성하는 방법이다. 이 방법은 간단한 일을 수행하기 위해서 장황한 문장을 작성해

야 하는 경우의 문제들을 해결한다. 실제 프로그램의 작성 코드의 양과 오류를 줄여준다.

2.2 코드 생성(Code Generating)

2.2.1 수동적 코드 생성

수동적 코드 생성은 기본적으로 몇 개의 입력에서 주어진 출력을 생성하는 매개변수화 된 템플릿으로 실행시간에 계산하기엔 비용이 많이 드는 참조 테이블 등과 같은 코드를 생성한다.

2.2.2 능동적 코드 생성

어떤 지식을 단 하나의 형태로 만들어 놓고 어플리케이션이 필요로 하는 온갖 형식으로 변환할 수 있게 한다. 예를 들어 스키마를 통해서 구조체를 만드는 경우 스키마가 변경될 때마다 그 스키마에 접근하기 위해 사용되는 코드 또한 자동으로 변경된다.

제안하고 있는 분산객체 통신 모듈은 코드 생성기를 통하여 매개변수로 받은XML문서를 파싱하여 해당 모의기가 사용할 메시지 처리에 대한 코드를 생성시킨다또한 실행시간에 메시지 송수신 여부가 작성된 문서를 입력받아 동적으로 함수를 실행시킬 수 있는 기능을 작성함으로써 메타 프로그래밍 기법을 적용한다.

3. 아키텍처

본 장에서는 기존에 연구된 시뮬레이션 네트워크 관리자에서 발생된 문제점을 해결하기 위한 방법으로 메타 프로그래밍의 코드 생성 방법을 적용한 분산객체 통신 모듈의 아키텍처를 제안하고 주요 구성요소들에 대한 특징과 기능을 알아본다.

제안하는 시뮬레이션 네트워크 관리자의 아키텍처는 그림 5와 같다.

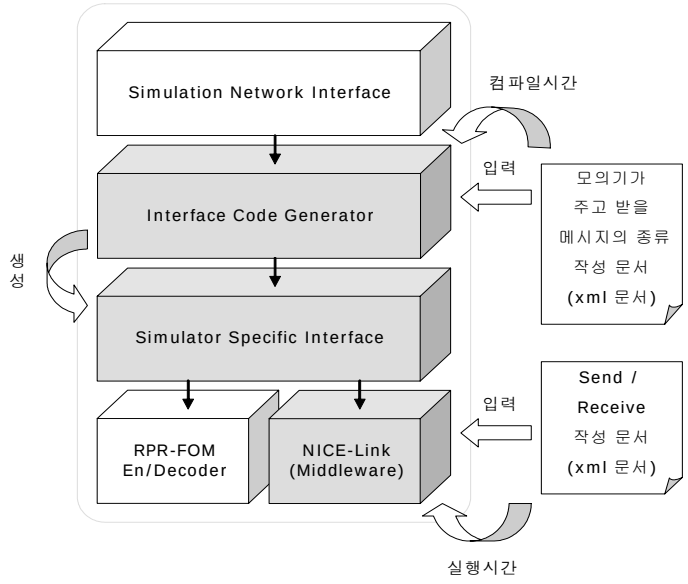

그림 5 시뮬레이션 네트워크 관리자 구조

3.1 구성요소

3.1.1 Simulation Network Interface

기존의 Simulation Network Interface 모듈과 동일하게 모의기 내 관리자들이 사용하는 내부 메시지와 모의기 간에 사용하는 외부 메시지 사이의 적응자(Adapter) 역할을 하며, Interface Code Generator 모듈로부터 생성된 Simulator Specific Interface 모듈을 사용한다.

3.1.2 Interface Code Generator

모의기가 주고받을 메시지의 구조에 대하여XML형식으로 작성된 문서를 입력 받는다. 이 모듈은 문서를 파싱하여 해당 모의기가 주고받을 메시지 처리 관련 코드를 생성시킨다. 즉, 정보를 더 단순하고 언어에 중립적인 형태로 표현해 놓은 다음 특정한 언어의 코드로 생성시킴으로써 모의 환경에 존재하는 모의기 들이 공통 인터페이스 모듈의 변경으로부터 유연성을 얻게 된다.

Interface Code Generator 모듈의 기능 구조를 살펴보면 그림 6과 같다.

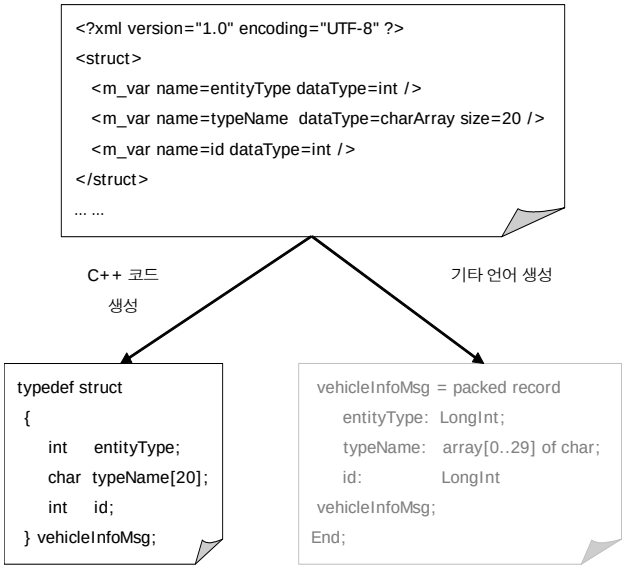

그림 6 Interface Code Generator 모듈의 기능

XML파일은 실제 모듈의 구현 언어 외에도 다른 언어를 생성해내는 코드를 작성하기에도 간단한 파서를 작성하여 코드 생성이 용이하다.

3.1.3 Simulation Specific Interface

Interface Code Generator로부터 생성된 해당 모의기 고유의 인터페이스 모듈로써Encoder / Decoder 모듈과 미들웨어 모듈인 NICE-Link를 사용하여 외부와의 통신 기능을 담당한다.

3.1.4 RPR-FOM En/Decoder

RPR-FOM(Real Time Platform - Federation Object Model) En/Decoder 모듈은 RPR-FOM의 정보들에 대해서 인코딩하거나 디코딩하는 모듈이다. RPR-FOM[5]은

해당 모의기가 주고받을 정보에 대한 데이터 구조를 기술해 놓은 표준이다.

3.1.5 NICE-Link

RTI의 사용자 편의성을 구현한 모듈로 실제로 인코딩 또는 디코딩된 정보를 RTI의 서비스를 사용하여 주고받는 기능을 담당한다[6]. 이 모듈은 사용자로부터 모의기가 주고받을 메시지의 송수신 여부에 대한 XML형식의 문서를 입력받아 실행시간에 조건을 변경할 수 있다. 다시 말해서 해당 모의기가 최초에 송신 인터페이스 규약을 따르다가 사용자 문서에 의해 동적으로 수신 인터페이스로 변경이 가능하다는 것을 의미한다.

따라서 실제 인터페이스 코드는 추상화 시키고 세부 내용은 메타데이터 정보인 XML문서에 작성함으로써 기능에 대한 설정 가능 모듈을 구현하게 된다. 작성된 XML 문서의 구조는 그림 7과 같이 sharing(공유) 속성을 통해서 메시지 전송여부 방식을 판단한다.

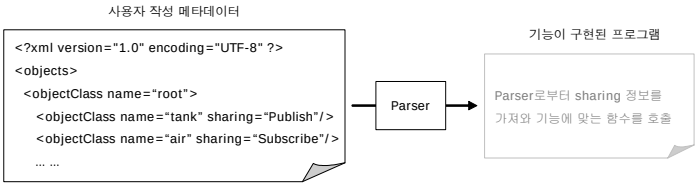

그림 7 메시지전송 방식에 대한 메타데이터 구조

3.2 모듈들 간의 협력관계

주요 구성요소들 간의 협력 관계를 시퀀스 다이어그램으로 살펴보면 다음과 같다.

3.2.1 모의기가 사용할 메시지 처리기능 생성

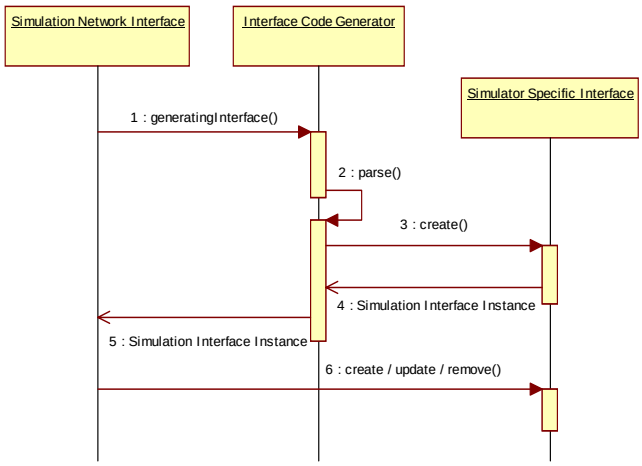

그림 8 메시지 처리기능 생성 시퀀스 다이어그램

3.2.2 메시지 송·수신 여부 변경 기능

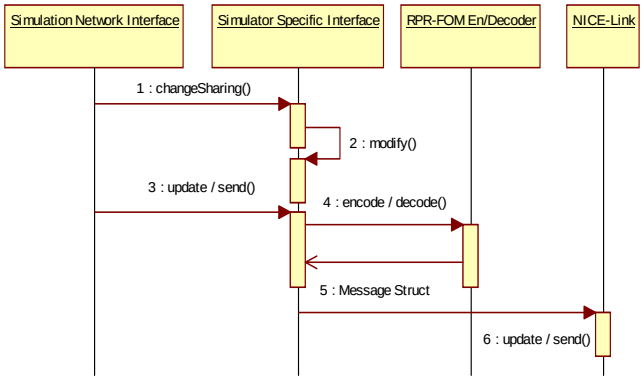

그림 9 메시지 송·수신 변경 시퀀스 다이어그램

기존의 Simulation Network Interface 모듈의 메시지 처리 기능은 Interface Code Generator 모듈로부터 생성된 Simulator Specific Interface 모듈로 위임되었다. 이 모듈은 미리 정의된 형식을 지원받아 생성되었기 때문에 시뮬레이션 네트워크 관리자를 사용하는 모의기마다 다른 구현을 소유하게 된다. 따라서 특정 모의기의 인터페이스에 대한 수정 요구사항이 발생 할 경우 변경의 범위를 특정 모의기로 한정시킴으로써 전체적인 구조의 깨짐 현상을 방지한다.

3.3 특징 및 장점

본 논문의 아키텍처는 세부사항 정보를 정의한 문서로부터 Interface Code Generator 모듈을 통하여 통합 형태의 코드가 아닌 설정 형태의 코드를 생성한다. 또한 시뮬레이션 네트워크 관리자를 사용하는 모의기가 "어떻게"가 아닌 "무엇을" 해야 하는지를 명시함으로써 선언적 프로그래밍을 지원한다.

제안한 접근법은 소프트웨어를 구현하고 변경 요구사항을 처리하는데 있어 다음과 같은 장점을 제공한다.

첫째, 코드의 추상화와 세부 내용의 분리를 통해서 설계의 결합도를 줄여 좀 더 유연하고 적응성 있는 프로그램을 생성할 수 있다. 이는 시뮬레이션 네트워크 관리자가 사용하는 메시지처리 방법을 추상 코드로 작성하고 메시지의 종류를 사용자 문서로 분리시킴으로써 발생하는 이점을 의미한다.

둘째, NICE-Link모듈의 경우 메시지 전송방식에 대한 문서를 실행시간에 설정할 수 있기 때문에 소프트웨어의 인터페이스 변경을 위한 다시 컴파일의 작업이 불필요하다.

셋째, XML형식과 같은 메타데이터 작성 방식은 범용적인 프로그래밍 언어보다는 문제 도메인에 가깝게 표현할 수 있기 때문에 언어 중립적인 해결책을 제시할 수 있다.

4. 결론 및 향후 연구방향

본 논문은 기존에 제안되었던 계층적 아키텍처 구조를

갖는 시뮬레이션 네트워크 관리자의 인터페이스 규약의 변경 시 발생하는 문제를 해결하기 위해서 메타프로그래밍 기법을 적용하였다. 제안한 아키텍처는 기존 모듈에 코드 생성기(Interface Coder Generator)가 추가되어 해당 관리자가 적용되는 각각의 모의기마다 특정 인터페이스를 정의하도록 지원한다. 또한 NICE-Link모듈이 실행 시간에 메시지 전송관련 정보를 설정 및 변경할 수 있게 하여 유연한 소프트웨어 모듈 디자인을 지향하고 있다.

현재 시뮬레이션 네트워크 관리자를 포함하고 있는 프레임워크의 프로토콜은 MOM(Message Oriented Middleware) 방식을 사용하고 있다. 즉, 프레임워크를 구성하는 각 관리자들이 사용하는 메시지 구조가 동일하며, 정의된 메시지를 통해서 통신한다. 따라서 메타프로그래밍 방법의 적용이 제한적일 수밖에 없으며 제안한 아키텍처의 이점을 충분히 얻을 수 없다.

따라서 메시지 기반 프로토콜에 대한 메타프로그래밍 적용 방안의 연구를 통해 제안 모듈의 장점을 극대화할 수 있어야 한다. 또한 코드 생성 시 제약사항이 될 수 있는 요소들을 파악해서 일반화 시킬 수 있는 방안을 모색해야 한다.

5. 참고문헌

[1] IEEE, "IEEE Standard for Modeling and Simulation(M&S) High Level Architecture (HLA) - Federate Interface Specification." IEEE Standard No.: 1516.1 - 2000

[2] 심준용, 진정훈, 김세환, "M&S Framework를 적용한 효율적인 분산객체 통신모듈 설계, 한국소프트웨어공학회 학술대회 논문집 제10권 제1호, 208p, 2008

[3] 전병선, ".NET Enterprise System 객체지향 CBD 개발 방법론", 2004

[4] http://www.ibm.com/developerworks/kr/library/l-metaprog1.html

[5] Simulation Interoperability Standards Organization Inc., "RPR-FOM Version 1.0 SISO-STD-001.1-1999", 1999.

[6] 심준용, 진정훈, 김세환, "분산객체 통신 시스템의 효율적인 개발을 위한 미들웨어, 한국정보처리학회 논문집 제14권 제3호, 245p, 2007