한국시뮬레이션학회 논문지
Vol. 20, No. 4, pp. 97-104 (2011. 12)

HLA기반 시뮬레이션 소프트웨어 개발을 위한 분산객체 통신 프레임워크 설계 및 구현

심준용[1†] · 위성혁[1] · 김세환[1]

Design and Implementation of Distributed Object Communication Framework for the Development of Simulation Software based HLA

Junyong Shim · Sounghyouk Wi · Saehwan Kim

ABSTRACT

Defense M&S software industry has carried out a variety of studies related to an efficient implementation of large-scale simulation and interoperability with respect to each of the system and HLA has been developed to provide a common architecture for distributed simulation of them. HLA defines Federate interface specification and provides services through RTI. Meanwhile, the difficulty lies in developing the software based HLA. Federate developer needs to understand how to handle Metadata produced RTI and has to modify the interface code whenever FDD is modified. This paper presents the implementation method of SOM interface using the code generation technique and middleware architecture for providing simple API. It solves the problem for implementing the framework of distributed object communication by using proposed method.

Key words : RTI, SOM, Framework Design, Communication Middleware, XML

요 약

국방 M&S의 소프트웨어 분야는 대규모 시뮬레이션의 효율적인 수행 및 각 시스템 사이의 상호 연동과 관련하여 다양한 연구를 수행해 왔으며, 그 중 HLA는 분산 시뮬레이션의 공통 아키텍처 표준으로 자리 잡고 있다. HLA는 Federate 인터페이스 규격을 정의하고, 구현물인 RTI를 통해 서비스를 제공한다. 한편, HLA기반 소프트웨어 개발은 몇 가지 어려움이 있다. 먼저 Federate 개발자는 RTI로부터 생성되는 메타 정보들을 다루는데 익숙해야 한다. 또한 동일한 FDD를 통해 Federate의 연동 인터페이스를 구현하기 때문에 FDD가 변경될 때마다 코드를 수정해야 한다. 본 논문은 코드 생성 기술을 활용한 SOM 인터페이스의 구현 방법과 RTI 기능에 대한 단순 API를 제공하는 미들웨어 구조를 제시하고, 제안 방법을 적용한 HLA기반의 분산객체 통신 프레임워크를 구현함으로써 앞에서 살펴본 문제를 해결하고자 한다.

주요어 : RTI, SOM, 프레임워크 설계, 통신 미들웨어, XML

1. 서 론

최근 국방 분야에서 이슈가 되고 있는 M&S(Modeling & Simulation)는 실제 시스템의 목적을 모의하기 위하여 구성요소, 모의절차 및 과정을 물리적, 수학적, 논리적으로 모델링하고, 산출물인 모델을 구현하여 시간의 흐름 속에서 실행하는 것으로써, 기존 워 게임(War Game) 영역을 확대하여 국방기획을 위한 소요제기, 무기획득, 결정, 분석평가 및 교육훈련에 이르기까지 과학화 지원을 위한 수단으로 활용된다[1]. 특히, 국방 M&S의 소프트웨어 분야는 대규모 시뮬레이션의 효율적인 수행 및 각 시스템 사이의 상호 연동과 관련하여 다양한 연구들을 수행해 왔으며, 그 중 HLA(High Level Architecture)는 분산 시뮬레이션을 위한 공통의 아키텍처를 제공하기 위한 기술 표준으로 자리 잡고 있다. HLA는 미 국방성(DoD)으로부터

접수일(2011년 9월 7일), 심사일(1차 : 2011년 12월 8일), 게재 확정일(2011년 12월 8일)
[1] LIG넥스원(주) Maritime연구소
주 저 자 : 심준용
교신저자 : 심준용
E-mail; jyshim79@lignex1.com

제안된 아키텍처로서 분산 시뮬레이션에 필요한 기술들을 정의함으로써 구성요소의 재사용(Reuse)과 상호운용성(Interoperability)을 확보하기 위한 표준이며, HLA 규칙(Rules)[2], Federate 인터페이스 규격(Interface Specification)[3] 및 객체 모델 템플릿(Object Model Template, 이하 OMT)[4]의 요소로 구성된다. HLA기반 시뮬레이션은 그림 1과 같이 Federate 인터페이스 규격의 구현물인 RTI(Run-Time Infrastructure) 소프트웨어를 사용하여 개발할 수 있으며, 다양한 시뮬레이션 체계와 상호연동 할 수 있다.

HLA기반 시뮬레이션은 모의 모델을 구현하고 있는 Federate와 실행 시간에 Federate 그룹의 정보 교환 환경을 제공하는 Federation으로 구성되며, RTI에서 제공하는 인터페이스를 구현하여 Federate 간 통신을 지원한다. RTI는 RTIambassador과 FederateAmbassador 클래스를 통해 사용자 API를 제공하며, 6개의 관리 서비스로 나누어 인터페이스를 기술하고 있다. RTIambassador는 RTI의 서비스 구현을 제공하고, FederateAmbassador는 RTI가 사용자 구현을 호출할 수 있도록 추상 클래스를 정의한다. 그림 2는 Federate와 RTI 간 인터페이스 구조를 보여준다. 특히, FederateAmbassador 클래스는 RTI의 정보를 수신할 수 있는 유일한 방법을 제공하기 때문에 Federate 개발자로부터 반드시 구현되어야 한다.

Federate는 OMT 표준으로 작성된 연동 객체 모델인 FOM(Federation Object Model)을 통해서 Federation에 가입된 Federate들 사이의 연동 인터페이스를 정의하며, 정의된 FOM은 FDD(FOM Document Data, IEEE 1516 표준)로 작성된다. 따라서 Federation에 가입을 원하는 모든 Federate는 동일한 FDD를 사용하여 인터페이스를 구현해야 하며, 특히 FDD에 기술된 객체 모델(ObjectClass 또는 InteractionClass)은 공유(Sharing) 속성을 설정하여 정보의 송신(Publish) 및 수신(Subscribe) 여부를 결정하므로 연동 인터페이스 설계 시 고려되어야 한다.

한편, Federate 개발자는 HLA기반 소프트웨어 개발 시 크게 두 가지 어려움을 겪는다. 첫째, RTI로부터 생성되는 Handle 정보 즉, InstanceHandle, AttributeHandle, DimensionHandle과 같은 메타 정보를 제어하는데 익숙해야 한다. 둘째, Federation의 연동 인터페이스가 수정되면 각 Federate의 인터페이스 코드를 수정하고 다시 빌드해야 한다. 특히, 대규모 시뮬레이션 환경에서 FDD의 변경은 Federate의 상호관계에 따라 많은 양의 코드 수정이 요구될 수 있다. 이러한 문제들은 Federate의 오류 발생 가능성 및 재사용 비용을 높이고, 소프트웨어에 대한 신뢰성을 떨어뜨릴 수 있다. 하지만 RTI 기능을 위한 단순 API 제공과 FDD 변경으로부터 Federate 구현 코드를 분리함으로써 해결할 수 있다.

본 논문은 코드 생성 기술을 활용한 SOM(Simulation Object Model) 인터페이스의 구현 방법과 RTI 기능에 대한 단순 API를 제공하는 미들웨어 구조를 제시하고, 제안 방법을 적용한 HLA기반의 분산객체 통신 프레임워크를 구현함으로써 앞에서 살펴본 문제를 해결하고자 한다. 특히, Federate의 개별 인터페이스를 나타내는 SOM을 XML (eXtensible Markup Language) 형태로 작성하고, SOM에 기술된 객체 모델을 관리할 수 있는 래퍼 객체(Wrapper Object)를 정의함으로써 FDD 인터페이스의 구현을 자동화하는 방법을 보여준다. 또한 제안 프레임워크를 적용한 시뮬레이션 연동 소프트웨어 개발 예제를 통해 프레임워크의 활용성을 보여준다. 논문의 구성은 다음과 같다. 2장은 HLA/RTI의 개요와 FDD를 이용한 시뮬레이션 연동 인터페이스 구현 방법을 소개한다. 3장은 SOM의 활용 방안에 대한 연구로서 SOM 작성법과 SOM 기반 연동 인터페이스의 구성 모듈인 SOM 파일 해석 모듈과 RTI 래퍼 객체 모듈의 구조를 살펴본다. 4장은 제안 방법을 적

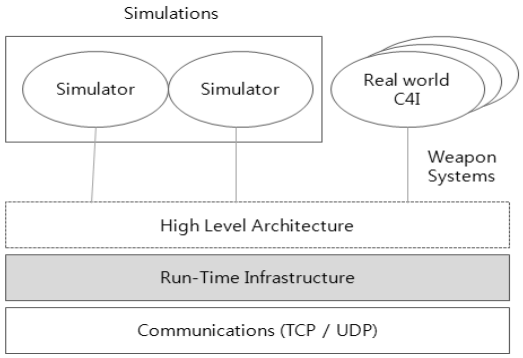

그림 1. HLA / RTI 구조

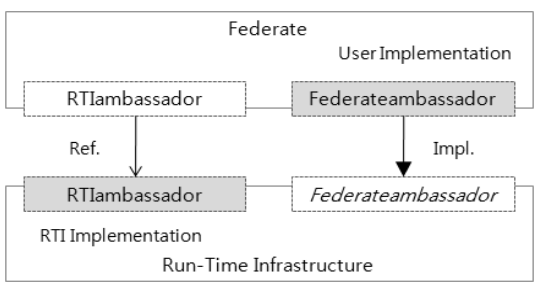

그림 2. Federate와 RTI 인터페이스 구조

용하여 HLA기반 분산객체 통신 프레임워크를 설계하고 구현한다. 5장은 구현된 프레임워크를 통해서 HLA기반 시뮬레이션의 연동 소프트웨어 개발 방법을 알아본다. 마지막으로 6장에서 결론을 맺고 향후 과제를 제시한다.

2. HLA/RTI 시뮬레이션

2.1 HLA/RTI 개요

HLA 표준은 분산 시뮬레이션의 공통 구조를 제시하기 위해 연합관리, 선언관리, 객체관리, 소유권관리, 시간관리 및 데이터 분산관리 서비스를 정의한다. 각 서비스가 제공하는 기능은 다음과 같다.

- 연합관리(Federation Management): Federation의 생성 및 삭제, 동기화, 저장 및 복구와 관련된 기능을 정의한다.
- 선언관리(Declaration Management): Federation에 가입된 Federate 사이의 교환 메시지에 대한 공유 방식을 선언하기 위한 기능을 정의한다.
- 객체관리(Object Management): ObjectClass의 Instance를 등록, 수정 및 삭제하고, InteractionClass를 송신 및 수신할 수 있는 기능을 정의한다.
- 소유권관리(Ownership Management): Instance의 Attribute에 대한 소유권 이전(이양 및 획득)과 관련된 기능을 정의한다.
- 시간관리(Time Management): Federation 내에서 전달되는 메시지의 인과성(Causality)을 보장하기 위한 메커니즘을 제공하며 관련 기능을 정의한다.
- 데이터 분산관리(Data Distribution Management): 영역(Region)을 사용하여 불필요한 데이터의 전송 및 수신을 줄이기 위한 기능을 정의한다.

각 기능은 Federate 인터페이스 표준 구현물인 RTI를 통해 API로 제공된다. RTI는 RTIExec(RTI Executive Process), FEDExec(Federation Executive Process) 및 LRC(Local RTI Component)의 구성요소를 통해 데이터 교환 및 시간 동기화를 수행한다. 데이터 교환을 위해 선언관리, 객체관리 및 데이터 분산관리를 사용하며, 시간 동기화는 시간관리 서비스를 사용한다.

2.2 FDD를 이용한 연동 인터페이스 구현

Federate의 연동 인터페이스는 FDD를 기반으로 작성되며, ObjectClass와 InteractionClass를 사용하여 정보 교환을 수행한다. 그림 3은 FDD 파일의 예제이다.

RTI는 Publish-Subscribe 방식[5]의 연동 메커니즘을

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE objectModel SYSTEM "HLA.dtd">
<objectModel other="Created with Visual OMT 1516" date="2003-06-11" version="1.0" type="FO
 - <objects>
    - <objectClass name="HLAobjectRoot" sharing="Neither">
        <attribute name="HLAprivilegeToDeleteObject" sharing="PublishSubscribe" order="
          dimensions="NA" updateType="NA" dataType="NA"/>
       + <objectClass name="A" sharing="PublishSubscribe" semantics="Depict global, spatia
       + <objectClass name="C" sharing="PublishSubscribe" semantics="Depict global, spatia
       - <objectClass name="D" sharing="PublishSubscribe" semantics="Something">
          <attribute name="da" sharing="PublishSubscribe" order="Receive" transportation=
            updateType="Conditional" dataType="int" semantics=""/>
        </objectClass>
       + <objectClass name="HLAmanager" sharing="Neither">
      </objectClass>
  </objects>
 - <interactions>
    - <interactionClass name="HLAinteractionRoot" sharing="Neither" order="TimeStamp" trar
      - <interactionClass name="X" sharing="PublishSubscribe" order="Receive" transportatio
          <parameter name="xa" dataType="RTIObjectIdStruct" semantics="" nameNotes="
          <parameter name="xb" dataType="SupplyStructArray1" semantics=""/>
          <parameter name="xc" dataType="RTIObjectIdStruct" semantics="" nameNotes="
```

그림 3. FDD 파일 예제

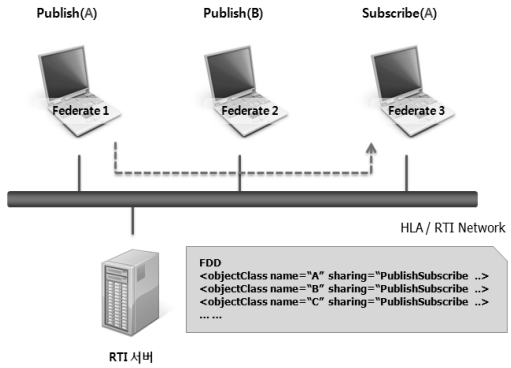

그림 4. RTI 데이터 교환 방법

제공하므로 Federate 사이의 통신은 관심 Federate가 아닌 관심 정보를 중심으로 수행된다. 즉, 그림 4와 같이 객체 모델 A를 "Publish"한 경우 해당 객체를 "Subscribe"한 Federate만 해당 정보를 수신할 수 있다.

FDD는 Federation에 존재하는 모든 Federate들의 관심 객체 및 공유 정보를 포함하며, 개별 Federate의 연동 인터페이스 구현 시 참조된다. 따라서 Federate의 인터페이스가 변경된다면 FDD와 구현 코드가 수정되어야 한다.

3. SOM기반 연동 인터페이스

Federation을 설계할 때 각 Federate의 인터페이스를 정의하기 위해 SOM을 작성하며, 모든 Federate의 SOM을 통합하여 FDD 파일을 작성한다. FDD는 RTI의 연합관리 서비스인 *createFederationExecution()* 함수를 통해 Federation 생성 및 가입된 Federate들의 인터페이스 구현을 위한 객체 모델의 Handle을 생성하는데 사용된다. 만약 개별 Federate의 객체 모델 정보가 FDD와 같은 형

태인 XML로 작성되어 있다면, XML 해석(Parsing)을 기반으로 한 코드 생성기법[6]을 이용하여 인터페이스 코드를 자동 생성할 수 있다. 특히, 객체 모델의 공유 속성을 읽어서 선언관리 서비스인 *publisObjectClassAttributes()*와 *subscribeObjectClassAttributes()*의 수행을 자동화 할 수 있기 때문에 Federate의 교환 객체가 자주 변경될 경우 유용하다. 이번 장은 SOM기반 연동 인터페이스 구현을 위한 설계 구조와 각 구성요소의 기능을 살펴본다.

3.1 SOM 작성 방법

SOM을 생성하기 위해 먼저 SOM 테이블을 작성한다. 예를 들어, 임의의 Federation에 2개의 Federate가 가입되어 있다면 그림 5와 같이 SOM 테이블을 구성할 수 있다.

SOM 테이블이 작성되면 FDD와 동일한 스키마 파일을 통해 XML을 생성한다. 그림 6은 ObjectClass A, B를 갖는 Federate 1의 SOM 파일, ObjectClass A, C를 갖는 Federate B의 SOM 파일 및 ObjectClass A, B, C를 갖는 FDD 파일의 작성된 구조를 보여준다.

SOM은 FDD와 다르게 각 Federate에 필요한 객체 모델을 가지며, 공유 속성 외에도 영역 정보(Dimensions), 전송 타입(Transportation Type), 순서 타입(Order Type)과 같은 스키마에 정의된 정보들을 포함한다. 논문에서 제안하는 SOM 파일 해석 모듈은 인터페이스 생성과 관련하여 공유 속성과 영역 정보만을 다룬다.

Federate 1

	Publish	Subscribe	Dimension	...
ObjectClass A				
Attribute aa	O		D1	...
Attribute ab	O		D1	...
ObjectClass B				
Attribute ba	O	O	D2	...
Attribute bb	O	O	D2	...

Federate 2

	Publish	Subscribe	Dimension	...
ObjectClass A				
Attribute aa		O	D1	...
Attribute ab		O	D1	...
ObjectClass C				
Attribute ca	O		D3	...
Attribute cb	O		D3	...

그림 5. SOM 작성 예

Federate 1

```
<objects>
    <objectClass name="ObjectClassA" sharing="Publish" dimensions="D1" ...>
        <Attribute name="aa" sharing="Publish" dimensions="D1" .../>
        <Attribute name="ab" sharing="Publish" dimensions="D1" .../>
    <objectClass name="ObjectClassB" sharing="PublishSubscribe" dimensions="D2" ...>
        <Attribute name="ba" sharing="PublishSubscribe" dimensions="D2" .../>
        <Attribute name="bb" sharing="PublishSubscribe" dimensions="D2" .../>
```

Federate 2

```
<objects>
    <objectClass name="ObjectClassA" sharing="Subscribe" dimensions="D1" ...>
        <Attribute name="aa" sharing="Subscribe" dimensions="D1" .../>
        <Attribute name="ab" sharing="Subscribe" dimensions="D1" .../>
    <objectClass name="ObjectClassC" sharing="Publish" dimensions="D3" ...>
        <Attribute name="ca" sharing="Publish" dimensions="D3" .../>
        <Attribute name="cb" sharing="Publish" dimensions="D3" .../>
```

FDD

```
<objectModel other="Created with Nex1" date="2011-08-17" ...
  <objects>
    <objectClass name="HLAobjectRoot" sharing="Neither">
      <objectClass name="ObjectClassA" sharing="PublishSubscribe" dimensions="D1" ...>
        <Attribute name="aa" sharing="PublishSubscribe" dimensions="D1" .../>
        <Attribute name="ab" sharing="PublishSubscribe" dimensions="D1" .../>
      <objectClass name="ObjectClassB" sharing="PublishSubscribe" dimensions="D2" ...>
        <Attribute name="ba" sharing="PublishSubscribe" dimensions="D2" .../>
        <Attribute name="bb" sharing="PublishSubscribe" dimensions="D2" .../>
      <objectClass name="ObjectClassC" sharing="Publish" dimensions="D3" ...>
        <Attribute name="ca" sharing="Publish" dimensions="D3" .../>
        <Attribute name="cb" sharing="Publish" dimensions="D3" .../>
```

그림 6. 개별 SOM과 FDD 파일 예제

3.2 SOM기반 연동 인터페이스 구조

SOM기반 연동 인터페이스는 그림 7과 같이 SOM 해석 모듈과 RTI 래퍼 객체 모듈로 구성된다.

SOM 파일 해석 모듈은 SOM 파일을 해석하고 메타정보를 객체화하는 기능을 수행하며, RTI 래퍼 객체 모듈은 LRC와 FederateAmbassador를 구현하여 RTI 기능을 제공한다. 각 모듈의 기능을 살펴보겠다.

3.2.1 SOM 파일 해석 모듈

Federate의 인터페이스 구현은 RTI의 선언관리 서비스 중 Publish와 Subscribe 기능에 대한 구현을 의미하며, 해당 기능을 통해 그림 8과 같이 상대 Federate와 교환할 객체 모델의 공유 정보를 생성하는 것이다. RTI는 이러한 정보를 기반으로 PS(Publish-Subscribe) 라우팅 테이블을 생성한다.

그림 8의 예제에서 Federate 개발자는 FDD를 참조하여 해당 Federate에 맞는 선언관리 기능을 직접 구현해야 한다. 하지만 그림 9와 같이 XML 형태의 SOM을 해석하

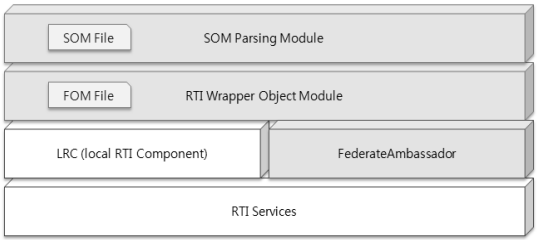

그림 7. SOM기반 연동 인터페이스 구조

고, 해석된 객체 모델의 정보들을 관리할 수 있다면 인터페이스 구현으로부터 사용자 코드를 분리할 수 있다.

SOM 파일 해석 모듈은 그림 10과 같이 ObjectClass, Attribute, InteractionClass 및 Parameter를 관리할 수 있도록 각각의 정보에 대응되는 구성요소를 갖는다. 모듈의 각 구성요소에 대한 기능은 다음과 같다.

- SOM Parser: SOM 파일을 입력받아 해석하고, 메타 정보 관리를 위해 Metadata 객체에 할당한다.
- OCMetadata: ObjectClass의 Handle, Dimension, Name 등의 메타 정보 관리와 OCInstance 객체생성을 위한 Factory 기능을 수행한다.
- OCInstance: OCMetadata로부터 Instance를 생성하여 객체관리 서비스의 주요 기능들을 제공한다.
- ACMetadata: Attribute의 Handle, Dimension, Name 등의 메타 정보 관리와 ACInstance 객체생성을 위한 Factory 기능을 수행한다.
- ACInstance: ACMetadata로부터 Instance를 생성하여 Attribute의 정보에 접근할 수 있도록 한다.
- ICMetadata: InteractionClass의 Handle, Name, Dimension 등의 메타정보 관리와 ICInstance 객체생성을 위한 Factory 기능을 수행한다.
- ICInstance: ICMetadata로부터 Instance를 생성하여 객체관리 서비스의 주요 인터페이스를 제공한다.
- PCMetadata: Parameter의 Handle, dimension, Name 등의 메타 정보 관리와 PCInstance 객체생성을 위한 Factory 기능을 수행한다.
- PCInstance: PCMetadata로부터 Instance를 생성하여 Parameter의 정보에 접근할 수 있도록 한다.

가령, 사용자가 ObjectClass의 Instance 생성을 요청할 경우 OCMetadata 객체는 OCInstance 객체를 생성하고, OCInstance에 포함된 Attribute 정보를 ACMetadata 객체로부터 가져온다. 값을 변경하기 위해서 ACInstance 객체를 생성하여 해당 값을 설정한다.

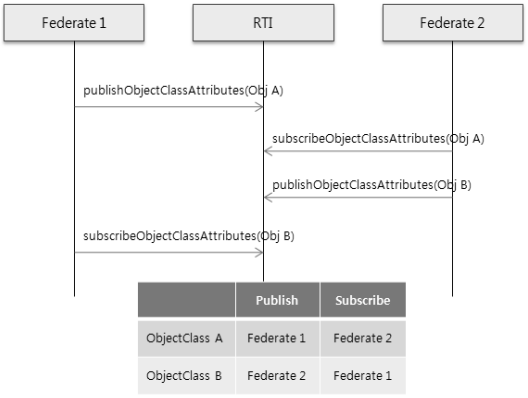

그림 8. 선언관리 서비스의 Publish / Subscribe 예제

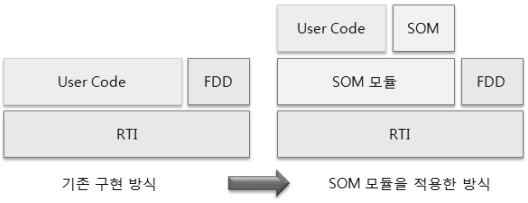

그림 9. SOM 파일 해석 모듈 적용 방법

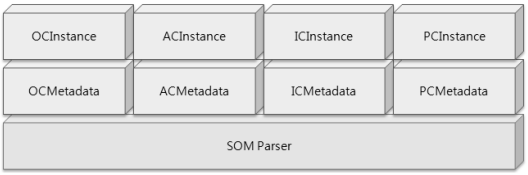

그림 10. SOM 모듈 구조

3.2.2 RTI 래퍼 객체 모듈

RTI 서비스를 제공하는 래퍼 객체 모듈은 메타 정보를 숨기고 문자열 중심의 단순 API를 제공한다. LRC에서 제공하는 RTIambassador와 FederateAmbassador를 구현하는 객체들로 구성된다. 그림 11은 RTI 래퍼 객체 모듈의 구조를 보여준다. RTIambassador를 관리하는 NICELink 모듈 외에 3개의 구성요소를 정의한다. 모듈의 구성요소에 대한 기능은 다음과 같다.

- NICELinkNet: RTIambassadorFactory로부터 RTI 서비스 사용을 위한 인터페이스를 생성한다.
- GlobalExercise: Federation 생성 및 삭제를 위한 RTI 기능을 구현하며, Federate를 관리한다.
- GEParticipant: Federate 가입 및 탈퇴를 위한 RTI 기능을 구현하며, 각 객체 모델을 관리하며, 정보를 수신하기 위해 INotificationManager를 구현한다.
- NotificationManager: INotificationManager를 구현하여 RTI로부터 수신되는 정보를 GEParticipant 객

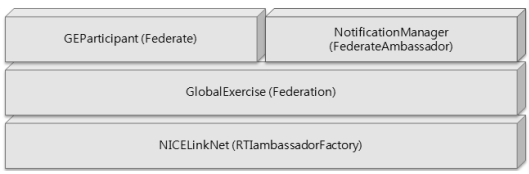

그림 11. RTI 래퍼 객체 모듈 구조

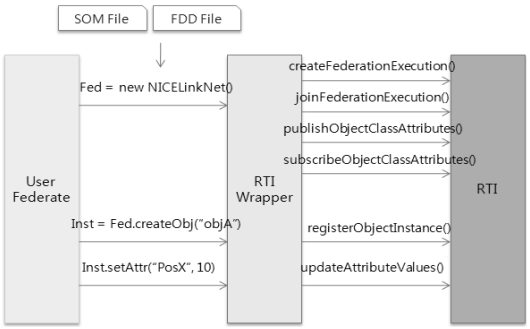

그림 12. 객체 지향 방식 적용

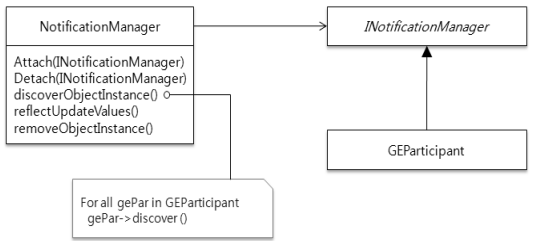

그림 13. Callback 메커니즘

체에게 전달한다.

· INoificationManager: GEParticipant가 Callback을 수신하기 위해 NotificationManager와 연결을 위한 추상 클래스를 제공한다.

RTI 래퍼 객체 모듈은 객체 지향 방식의 단순 API를 제공한다. 그림 12는 래퍼 객체 모듈의 사용 방법을 보여준다. Instance가 생성되면 등록-발견 구조에 의해 Subscribe를 선언한 Federate가 Instance를 발견한다. createObj() 함수는 지역(Local) 객체와 원격(Remote) 객체를 생성하여 이 구조를 지원한다.

GEParticipant와 NotificationManager 객체는 GoF패턴[7] 중 Observer 패턴을 적용하여 그림 13과 같이 RTI 수신 정보에 대한 Callback 메커니즘을 구현한다.

Subscribe를 선언한 Federate는 discover() 함수와 마찬가지로 FederateAmbassador의 Callback 함수를 통해 객체 정보의 값을 전달받는다. discover()는 생성 객체를 발견할 때, reflect()는 변경된 객체 정보를 수신할 때, remove()는 발견 객체를 제거할 때 사용된다.

4. 분산객체 통신 프레임워크

본 장은 SOM기반 연동 인터페이스를 적용하여 분산객체 통신 프레임워크를 제안한다. 제안 프레임워크는 HLA 기반 시뮬레이션의 연동 망 구현 시 재사용 컴포넌트를 제공하며, RPR(Real-Time Platform Reference) FOM[8] 기반의 데이터 연동을 지원한다. 프레임워크의 구조는 그림 14와 같다.

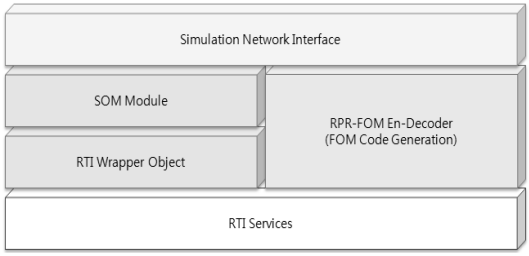

그림 14. 분산객체 통신 프레임워크 구조

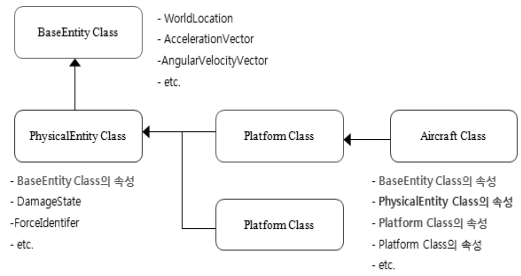

그림 15. RPR-FOM의 ObjectClass 구조 예제

• SimulationNetworkInterface 모듈

사용자 요구 메시지와 RTI 객체 간 적응자(Adapter) 역할을 수행하며, SOM기반 연동 인터페이스 모듈과 RORFOM En-Decoder 모듈을 통합한다.

• SOM기반 연동 인터페이스 모듈

SOM 모듈과 RTI 래퍼 객체 모듈로 구성되며, 각 내용은 3장에서 기술했다.

• RPR-FOM En-Decoder 모듈

네트워크를 통해 교환되는 객체는 객체가 갖는 속성을 바이트 스트림으로 변환하는 Encoding과 바이트 스트림을 객체로 변환하는 Decoding이 필요하다. RPR-FOM En-Decoder 모듈은 RPR-FOM에 정의된 객체 모델의 속성 정보를 위한 Encoding 및 Decoding 기능을 제공한다. RPR-FOM은 Federate들 사이의 의미적(Semantic) 상호연동을 위해 SISO(Simulation Interoperability Standard Organization)로부터 제정되었다. RPR-FOM에 정의된 ObjectClass의 구조는 그림 15와 같다.

그림 16은 RPR-FOM En-Decoder 모듈에서 제공하는

```
// WorldLocation
bufWL = new char[ worldLocation.GetSize() ];
memset( bufWL, 0, worldLocation.GetSize() );
worldLocation.Encode( bufWL, worldLocation.GetSize() );
stEndeCodeInfo.name = "WorldLocation";
stEndeCodeInfo.buf = bufWL;
stEndeCodeInfo.size = worldLocation.GetSize();
msgList.push_back( stEndeCodeInfo );

// AccelerationVector
bufAV = new char[ accelerationVec.GetSize() ];
memset( bufAV, 0, accelerationVec.GetSize() );
accelerationVec.Encode( bufAV, accelerationVec.GetSize() );
stEndeCodeInfo.name = "AccelerationVector";
stEndeCodeInfo.buf = bufAV;
stEndeCodeInfo.size = accelerationVec.GetSize();
msgList.push_back( stEndeCodeInfo );
```

그림 16. Aircraft 객체 Encoding 예제

Aircraft 객체의 WorldLocation과 AccelerationVector 속성에 대한 Encoding 기능의 구현 예제이다.

RPR-FOM En-Decoder는 FOM 코드 생성기를 통해서 생성된다[9]. 즉, 사용자 정의 FOM을 사용할 경우 코드 생성기를 통해 해당 FOM에 맞는 En-Decoder를 생성하여 모듈을 교체할 수 있다.

5. 적용 방법

제안 프레임워크는 HLA기반 시뮬레이션 연동 망 구현에 적용될 수 있다. 적용 시스템의 전체 구조와 연동 망의 통신 구조는 그림 17과 같다.

Federate 개발자는 FDD로부터 개별 인터페이스 목록을 생성하여 SOM 파일을 작성한다. SOM과 FDD 파일은 각각 SOM 모듈과 RTI 래퍼 객체 모듈에 입력되며, 해당 Federate가 Federation 가입과 동시에 SOM에 작성된 모든 정보를 해석하여 Publish 및 Subscribe 선언을 수행한다. 만약, 임의의 Federate가 사용하는 객체 모델의 공유 정보를 Publish에서 Subscribe로 변경할 경우 SOM 파일을 수정한 후 Federation에 재가입하면 된다. 제안 프레임워크의 핵심인 SOM기반 연동 인터페이스는 개발자가 작성할 코드를 자동으로 생성 및 호출해줌으로써 개발 소프트웨어의 신뢰성 및 유지보수성을 높인다.

6. 결론 및 향후과제

HLA는 국방 M&S의 소프트웨어 분야에서 분산 시뮬레이션을 위한 공통의 아키텍처를 제공하기 위한 기술 표준으로 자리 잡고 있다. 특히, HLA 표준 서비스의 구현물인 RTI는 분산 시뮬레이션 시스템 개발의 핵심 기술로워 게임, 훈련장비 및 가상 시뮬레이터 개발과 같은 다양한 사업에 적용되고 있다. 한편, Federate 개발자는 RTI

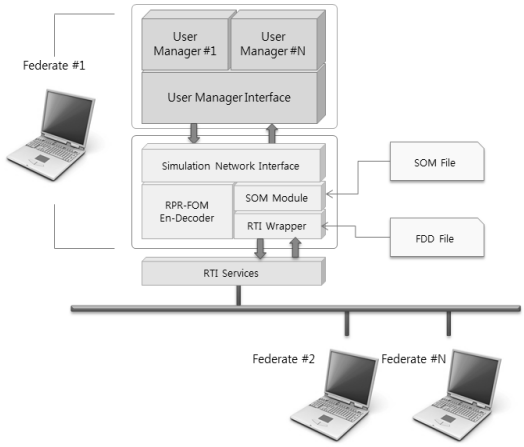

그림 17. HLA 서비스 망 적용 구조

를 사용하는데 있어 Handle과 같은 메타 정보를 다루는데 익숙해야 하며, 연동 인터페이스 코드가 FDD에 종속되므로 FDD 변경에 따라 코드 수정이 지속적으로 요구될 수 있다. 본 논문은 이러한 문제를 해결하기 위해서 SOM기반 연동 인터페이스 구조를 제시하고, SOM을 해석하는 기능과 RTI 기능을 단순화하는 모듈을 구현했다. 특히, SOM기반 연동 인터페이스를 적용한 분산객체 통신 프레임워크를 설계하여 HLA기반 시뮬레이션 소프트웨어 개발에 필요한 재사용 컴포넌트를 개발했다. 제안 프레임워크는 RPR-FOM 연동을 위한 개별 Federate의 인터페이스 코드를 자동으로 생성함으로써 개발 소프트웨어의 신뢰성 및 유지보수성을 제공할 것이다.

향후 제안 프레임워크는 개발 시스템의 요구 성능에 대한 만족 여부를 판단할 수 있도록 다양한 시나리오를 설정하여 처리량(Throughput) 및 지연시간(Latency) 측정을 수행해야 한다. 또한 시간관리 및 소유권관리 기능을 추가 구현함으로써 HLA의 모든 기능을 제공해야 한다.

참 고 문 헌

1. 방위사업청, “무기체계 획득단계별 M&S 적용지침”, 방위사업청 지침 제 2010-32호.
2. IEEE, “IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework an Rules.” IEEE Standard Mo: 1516-2000.
3. IEEE, “IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Federate Interface Specification.” IEEE Standard No.: 1516.1 - 2000
4. IEEE, “IEEE Standard for Modeling and Simulation

(M&S) High Level Architecture (HLA) - Object Model Template (OMT) Specification." IEEE Standard Mo: 1516.2-2000.
5. Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal, "Pattern-Oriented Software Architecture A System of Patterns", Wiley & sons, 1996.
6. Burginsky, F., M. Finnie, P. Yu, "Automatic Code Generation from Design Patterns", IBM Systems Journal, Volume 35, #2, 1996.
7. Gamma, E et.al. "Design Patterns: Elements of Reusable Object-Oriented Software", ISBN 0-201-63361-2
8. Simulation Interoperability Standards Organization Inc., "RPR-FOM Version 1.0 SISO-STD-001.1-1999", 1999.
9. 심준용, 진정훈, 김세환, "HLA기반 시뮬레이션의 연동 객체를 위한 코드 생성기", 정보통신분야학회 합동학술대회, 2009.

심 준 용 (jyshim79@lignex1.com)

2005 우석대학교 컴퓨터공학과 학사
2007 한양대학교 컴퓨터공학과 석사
2007～2009 M&S연구소 주임연구원
2009～현재 Maritime연구소 선임연구원

관심분야 : 모델링&시뮬레이션, HLA/RTI, 분산 컴퓨팅

위 성 혁 (wisounghyouk@lignex1.com)

2001 전남대학교 컴퓨터공학과 학사
2009 M&S연구소 선임연구원
2009～현재 Maritime연구소 선임연구원
2011 한남대학교 국방M&S학과 석사

관심분야 : SBA, 시스템 아키텍처, 모델링&시뮬레이션

김 세 환 (saehwan.kim@lignex1.com)

1985 경북대학교 전자공학과 학사
1987 경북대학교 전자공학과 석사
2005～현재 Maritime연구소 VM팀 팀장
2006～현재 국방과학기술조사 M&S자문위원
국방과학기술수준조사 기술전문가

관심분야 : 모델링&시뮬레이션, SBA, LVC