컴포넌트 재사용을 위한 DLL 플러그인 프레임워크 설계

심준용*, 이용헌*, 김세환*
*LIG넥스원(주) Maritime연구센터
e-mail:jyshim79@lignex1.com

A Design of DLL Plug-in Framework for Component Reuse

Jun-Yong Shim*, Yong-Heon Lee*, Sae-Hwan Kim*
*Maritime R&D Center, LIG Nex1 Co., Ltd.

요 약

최근 국방 소프트웨어 분야에서는 모델링 및 시뮬레이션 기술이 각광받으면서 무기체계 개발을 위한 시뮬레이션 소프트웨어 개발 사업을 늘리고 있다. 특히, 시뮬레이션 요소의 재사용성 및 신뢰성 확보를 위한 개발 프레임워크 제공이 핵심기술로 떠오르면서, 시뮬레이션을 위한 공통 서비스를 제공하는 M&S 프레임워크가 개발되었다. 하지만 고객의 요구사항이 프레임워크의 기능 변경을 요구하는 경우 프레임워크가 적용된 모든 시뮬레이션 요소의 수정이 불가피하며, 추가 구성요소의 상호작용을 위한 인터페이스 재설계가 요구된다. 본 논문은 이러한 문제점을 해결하기 위해서 프레임워크의 요소를 DLL로 구현하여 기능 구성을 용이하게 하고, 구성요소 간 상호작용을 위해 데이터 기반 Publish-Subscribe 방식을 사용함으로써 프레임워크와 독립적으로 인터페이스를 설계할 수 있도록 한다. 특히, 프레임워크와 DLL 간 교환 메시지 객체에 대한 구조 설계를 제시한다.

1. 서론

시스템의 외부 환경이 급변하고 소프트웨어 기술 요소 또한 복잡하고 다양해지는 상황에서 소프트웨어 시스템의 구축과 개선을 어떻게 하면 체계적이고 효율적으로 실행하느냐 하는 문제는 시스템 개발의 성공 여부를 좌우할 만큼 중요한 이슈가 되고 있다. 이러한 문제를 해결하기 위해서는 시스템 요구사항에 적합하고 다양한 품질 속성을 제공할 수 있는 소프트웨어 아키텍처 설계와 공통의 실행 기반을 제공하는 프레임워크 개발이 필요하며, 개발 시스템은 사용자 요구사항 변경에 대한 용이성과 구성 요소의 신뢰성 및 재사용성을 높일 수 있는 구조를 제공할 수 있어야 한다. 이러한 구조는 고객의 요구사항 변화에 빠르게 대처할 수 있도록 기능 재구성을 위한 유연성(Flexibility) 및 확장성(Extensibility)을 갖춰야 한다.

국방 소프트웨어 산업에서도 무기체계 또는 비 무기체계(정보체계) 개발 시 구성 요소의 재사용을 늘리고 체계 간 상호 연동을 제공하는 기술이 핵심이 되고 있다. 특히 국방 Modeling & Simulation(M&S)을 기반으로 하는 시스템 구축 시 개발 모델의 이식성을 높이고, 시뮬레이션의 상호 연동성을 높이기 위한 기술 표준으로 High Level Architecture(HLA)[1,2]를 적용함으로써 시스템의 다양한 품질 속성을 만족시키고 있다. 또한 HLA기반의 체계 시뮬레이션 개발에서 요구하는 다양한 품질 속성을 만족시키기 위한 노력의 일환으로 M&S 프레임워크가 개발되었으며[3,4], 멀티 쓰레드, 시간 스케줄링, 네트워킹과 관련된 공통의 서비스와 내부 모듈 간 메시지 교환을 위해 메시지 디스패처(Message Dispatcher)를 제공함으로써, 신뢰성, 재사용성 향상 및 비용절감을 가져왔다.

한편, 사용자는 프레임워크에서 제공하지 않는 기능을 요구하거나, 공통의 서비스를 사용하여 새로운 운영개념을 갖는 프레임워크를 요구할 수도 있다. 이러한 요구사항을 만족시키기 위해서 프레임워크를 구성하는 모듈 간 인터페이스는 느슨한 결합을 갖기 위한 프로토콜 설계가 고려되어야 하며, 동적으로 요구 기능을 추가하거나 제거할 수 있는 재구성 가능한 아키텍처를 제공해야 한다. 기존 M&S 프레임워크는 구성요소 간 메시지 디스패처를 중심으로 밀접하게 결합되어 있기 때문에, 기능 추가 시 인터페이스에 대한 재정의 및 프레임워크 수정이 불가피하다. 플러그인 아키텍처는 이러한 문제를 해결할 수 있는 소프트웨어 구조를 제시하며, 기존 시스템에 대한 기능 추가 및 제거에 의한 인터페이스 변경을 최소화시킬 수 있는 재구성 방법을 제공하고 있다[5].

본 논문은 DLL(Dynamic Linked Library) 기반의 플러그인 프레임워크 구조를 제안한다. 제안 프레임워크는 요구기능을 구현한 DLL을 프레임워크에 동적으로 재구성할 수 있으며, 데이터 기반 Publish- Subscribe 방식을 사용함으로써 메시지 교환을 위한 상호작용 인터페이스 설계가 용이하다. 구성은 다음과 같다. 2장은 기 개발된 M&S

프레임워크의 구조 및 특징을 살펴보고, 3장에서 M&S 프레임워크의 단점을 보완하기 위한 DLL 플러그인 프레임워크 구조를 제시한다. 4장과 5장은 제안 프레임워크의 구현 이슈와 성능비용을 분석하고, 마지막으로 5장에서 결론 및 향후 과제를 다룬다.

2. M&S 프레임워크

HLA 기반의 무기 체계 시뮬레이션 개발을 위한 M&S 프레임워크는 사용자 GUI와 상호작용을 위한 UI관리자, 구현 모델의 정보를 관리하는 모의 관리자, 원격 개체를 제어 및 감시하기위한 통제 관리자, 프레임워크 내부 관리자를 통제하는 설정통제 관리자, 실 장비와 연결된 링크 모의 개체 정보를 관리하는 파라미터 관리자, 시뮬레이션 시나리오를 관리하는 시나리오 관리자 그리고 RTI(Run Time Infrastructure)를 통해 분산 시뮬레이션 환경을 제공하는 시뮬레이션 관리자로 구성되어 있다. 그림 1은 프레임워크의 구조를 보여준다.

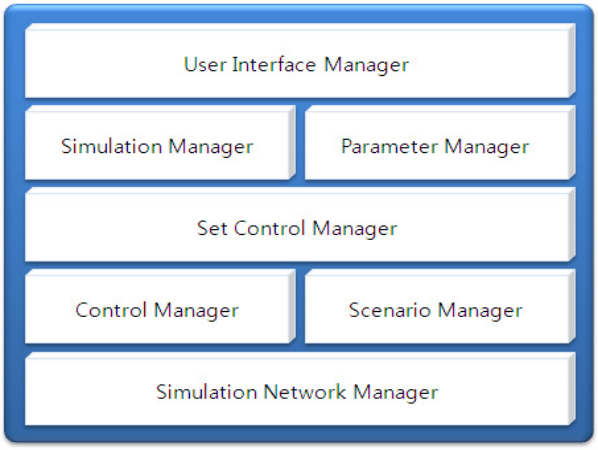

그림 1 M&S 프레임워크 구조

각 관리자는 내부적으로 메시지 기반 프로토콜인 메시지 디스패처를 사용하여 상호작용하고, HLA/RTI를 통해서 프레임워크 간 상호작용을 수행한다[그림 2].

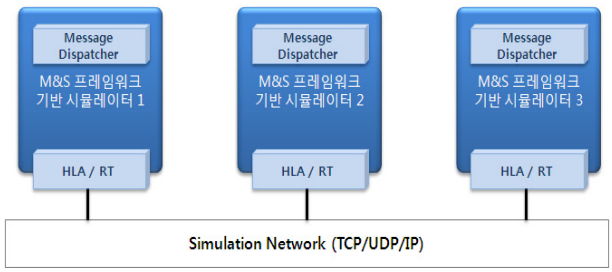

그림 2 프레임워크 상호작용 구조

프레임워크 구현에 핵심이 되는 메시지 디스패처는 Send()와 Receive() 서비스를 통해서 구성 요소 간 상호작용을 지원한다. 메시지 송신의 경우 수신 관리자의 식별자(ID)와 메시지 값(Msg)을 Send() 함수에 할당하고, 수신은 송신 관리자의 식별자와 전송된 메시지 값을 Receive() 함수에 할당하여 수행하게 된다. 메시지 디스패처를 사용 구조는 그림 3과 같다.

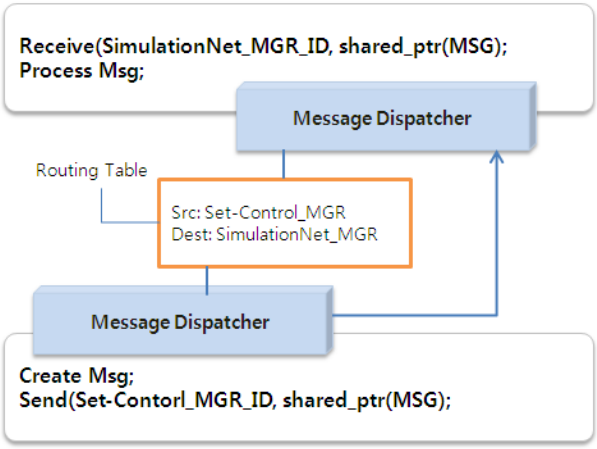

그림 3 메시지 디스패처 구조

메시지 디스패처는 공유 포인터를 사용하여 모듈 간 메시지 복사를 줄이고 메시지 포인터를 관리함으로써 잘못된 메모리 접근을 막아준다[6]. 따라서 각 관리자의 인터페이스를 구현할 경우 정의한 라우팅 테이블을 기반으로 메시지 교환을 위한 관리자를 식별하고 메시지 값을 설정하는 일에만 집중할 수 있다. 반면, 관리자의 식별자를 각 함수의 인자에 직접 입력하기 때문에 프레임워크의 작업 흐름이 변경되어 관리자가 바뀌거나 새로운 기능을 지원하기 위해 관리자를 추가해야 할 경우 기존 관리자들의 메시지 디스패처를 수정해야 한다. 특히, 설정 통제 관리자는 모든 관리자들의 메시지 경로 테이블과 관련되기 때문에 특정 관리자의 메시지 흐름이 변경될 경우 전체적인 흐름에 영향을 끼칠 수 있다. 따라서 재사용뿐만 아니라 인터페이스의 확장성 및 유연성을 제공하기 위해서는 다른 방식의 접근이 필요하다. 즉, 관리자 정보가 아닌 메시지 정보에 관심을 갖고 플러그 인 될 수 있도록 설계되어야 하며, 메시지 정보에 대한 정형화된 표현 형식이 정의되어야 한다.

3. DLL기반 플러그인 프레임워크

본 장에서 제안하는 DLL 기반 플러그인 프레임워크는 Publish-Subscribe 기반의 메시지 지향 미들웨어를 사용하여 구성 요소가 아닌 정보 단위의 상호작용이 가능하고, XML 기반의 메시지 정의를 통해서 작성 기능과 교환 기능을 분리한다. 따라서 기존 M&S프레임워크와 같이 고정된 구성요소가 아닌 DLL기반으로 재구성이 용이한 구조를 제공하며, 메시지 디스패처에 작성했던 라우팅 테이블을 XML 해석(Parsing)을 통해 동적으로 생성할 수 있다. 즉, 새로운 DLL 추가 시 상호작용 인터페이스에 대한 설

계가 용이해진다.

제안 프레임워크는 크게 구성요소 간 메시지 교환 서비스를 제공하는 MEB(Message Exchanging Bus)와 DLL 구현에 필요한 인터페이스를 제공하는 MEC(Message Exchanging Component)로 구성되어 있으며, Publish와 Subscribe를 담당하는 하부 모듈로 나뉘어 제공된다. 그림 4는 MEB와 MEC 컴포넌트의 구조를 보여준다.

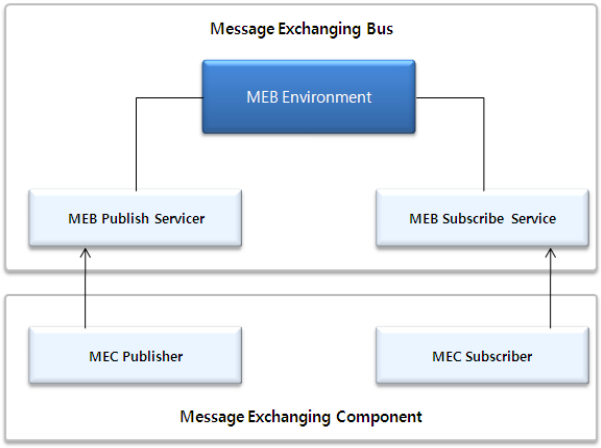

그림 4 프레임워크 컴포넌트 구조

MEC를 통해서 사용자 DLL을 구현할 수 있으며 그림 5와 같이 MEB에 플러그인 되어 상호작용 할 수 있다. 각 컴포넌트의 하부모듈인 Publish와 Subscribe는 메시지가 Send 또는 Receive 상태에 따라 참조된다.

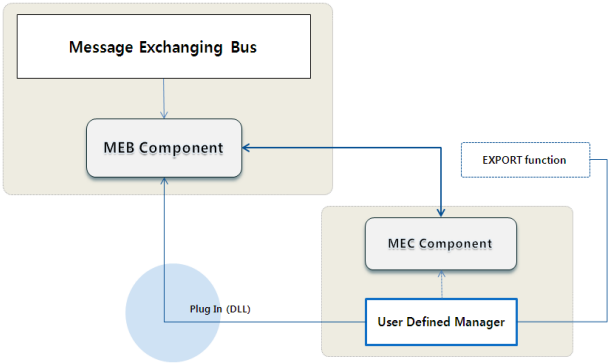

그림 5 MEB와 MEC 모듈 상호작용 구조

MEB는 MEC와 인터페이스를 위해서 DLL의 명시적 링크 방법[7]을 사용하는데 DLL 프로세스를 가상 메모리에 적재하는 LoadLibrary(), DLL 객체의 함수 포인터를 얻어오는 GetProcAddress(), DLL 해제를 위한 FreeLibrary()를 사용한다. 플러그인 구현을 위해 UserDefinedManager는 사용자 객체와 관련된 EXPORT함수(CreateInstance, DeleteInstance)를 정의해야 한다. 사용자 DLL이 MEB에 플러그인 되면 MEB의 생성 인스턴스를 사용자 객체에게 넘겨주고, 사용자는 할당된 MEB 객체를 통해서 Send()와 Receive() 서비스를 제공받을 수 있다.

4. 구현 이슈

플러그인 프레임워크는 사용자 DLL 구현 시 프레임워크의 구현 인터페이스와 독립적인 구조를 제공하는 것이 핵심이다. 즉, MEB의 어떠한 인터페이스에 관여하지 않고 MEC를 사용하여 사용자 DLL을 구현할 수 있어야 한다. 하지만 그림 6에서 볼 수 있듯이 MEC 객체가 MEB 객체의 함수를 링크 시에 참조하기 때문에 사용자 DLL은 MEB 객체를 생성하여 구현해야 하며, 이는 완전한 플러그인 시스템을 구현할 수 없게 한다.

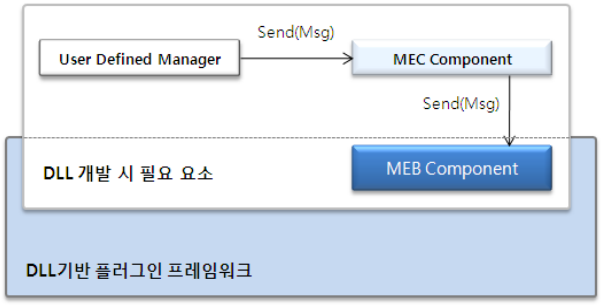

그림 6 플러그인 시 문제점

제안 프레임워크는 이러한 문제를 해결하기 위해서 그림 7과 같이 MEB의 상호작용 객체에 대한 서비스를 가상(Virtual) 함수로 제공하는 NullMEB 클래스를 정의했다. MEC 객체는 NullMEB 클래스를 참조하여 MEB 객체의 서비스를 사용한다. NullMEB의 함수들은 가상 테이블을 통해서 관리되며, 플러그인 후 각 서비스는 동적 바인딩에 의해 실제 MEB 객체를 참조하게 된다[8]. 즉, 사용자 DLL은 MEB와 독립적인 환경에서 구현할 수 있다.

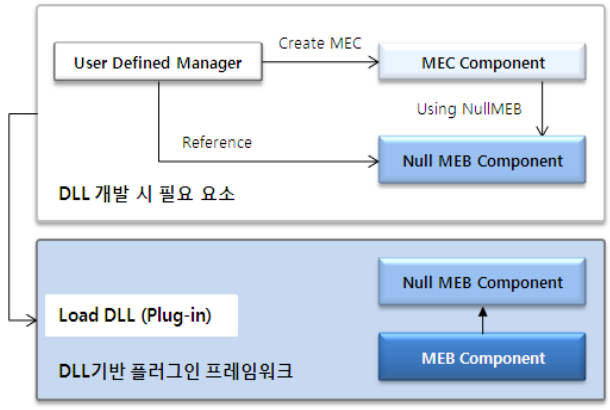

그림 7 NullMEB를 통한 결합 분리

이러한 구조를 갖기 위해서 MEC 객체는 MEB 객체의 함수를 사용하기 전에 프레임워크로부터 생성된 MEB 객체를 가져와야 한다. 즉, MEB 객체는 플러그인 DLL의 사용자 객체를 생성한 후, MEB 객체를 사용자 객체에게 할당해야 한다. 이러한 절차가 진행되면 동적 바인딩에 의해 기존의 프레임워크가 수행되었던 방식과 동일하게 서

비스에 접근하여 메시지를 교환할 수 있게 된다. 그림 8은 플러그인 인터페이스의 수행 절차를 보여준다.

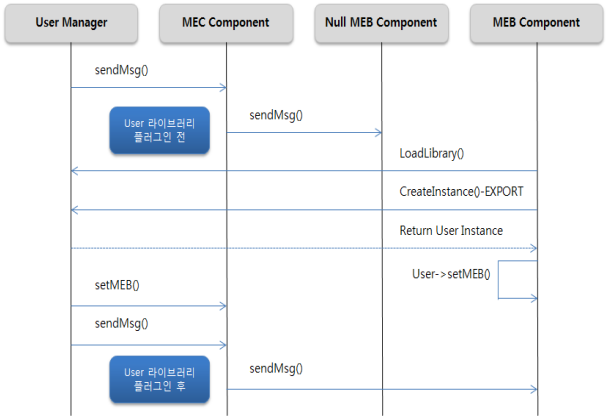

그림 8 플러그인 인터페이스 시퀀스

5. 성능비용

가상 함수로 이루어진 NullMEB 객체를 사용할 경우 동적 바인딩에 의해 발생할 수 있는 성능을 고려해야한다. 일반적으로 컴파일러가 가상 함수를 다루는 방법은 배열 형식의 가상 함수 테이블을 작성하여 가상 함수를 재구현한 파생 클래스의 함수 주소를 참조하는 방식이다. 제안 프레임워크의 경우 MEBComponent 클래스가 NullMEB Component 클래스를 상속받아 구현하고 있기 때문에 다음과 같은 코드 분석을 생각할 수 있다.

```
// 구현 코드
NullMEBComponent* meb;
meb = new MEBComponent;;

meb->func();

// 컴파일러 해석
(*meb->vptr[1])(meb);
```

컴파일러 해석에서 볼 수 있듯이 정적 바인딩을 사용한 경우보다 배열로 구성된 테이블 접근 및 인덱스 값 참조 비용이 추가된다. 한편, 메시지 송신 기능 제공이 목적인 NullMEB 객체는 가상 테이블에 Send() 함수 정도가 등록되므로 정적 바인딩과 비교하면 실제 함수 호출 전에 배열 접근 비용정도만 추가된다. 물론 부하시험(Load Test)을 통해 평가가 필요하지만 동적 바인딩을 통해 플러그인 구조를 얻은 것과 비교하면 무시할 수 있는 비용이다.

6. 결론 및 향후과제

국방 소프트웨어 분야에서도 다양한 IT 기술이 적용되고 있으며, 특히 재사용과 신뢰성 향상을 위한 프레임워크 개발이 다양한 무기체계 사업을 통해 이루어지고 있다. 한편, 시스템의 규모가 커지고 요구사항이 복잡해진다면 다수의 구성원이 참여하여 구현 및 통합을 수행해야 하기 때문에 프레임워크와 독립적으로 사용자 모듈을 구현할 수 있는 방법이 필요하다.

본 논문은 이러한 문제를 해결하기 위해서 DLL 기반의 플러그인 프레임워크 구조를 제시했다. 플러그인 구현을 위해서는 DLL의 명시적 연결 방법을 사용하는데 이때, 사용자 모듈과 플러그인 서버 모듈의 분리를 위해서 널(Null) 객체를 사용했다. 널(Null) 객체는 DLL의 개발 시점(컴파일 및 링크)까지 사용되다가 실행시간에 실제 객체로 교환되면서 플러그인 인터페이스를 제공하게 된다.

향후, 동적 기능 재구성을 위한 DLL의 추가 및 삽입 시 DLL 목록에 대한 관리 방법이 연구되어야 하며, 기 개발된 M&S 프레임워크의 구성 관리자들을 DLL로 구현함으로써 플러그 방식을 사용한 경우와 사용하지 않은 경우의 개발의 효율성 및 프레임워크 통합 모듈 간의 메시지 교환 성능의 비교 분석이 요구된다.

참고문헌

[1] IEEE, "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework an Rules." IEEE Standard Mo: 1516-2000.

[2] IEEE, "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification." IEEE Standard Mo: 1516.1-2000.

[3] 심준용, 진정훈, 김세환, "M&S Framework를 적용한 효율적인 분산객체 통신모듈 설계", 한국소프트웨어공학회 학술대회 논문집 제10권 1호, 2008

[4] 김성용, 윤근호, 김세환, 정태일, "데이터 링크 시뮬레이터 설계를 위한 M&S 프레임워크", 한국군사과학기술학회 종합학술대회 논문집, 2008

[5] Argent, R.M., "An overview of model integration for environmental applications- components, frameworks and semantics", Environmental Modeling & Software 19, 219-234, 2004

[6] http://www.boost.org/doc/libs/1_44_0/index.html

[7] http://msdn.microsoft.com

[8] Mark Deloura, "Game Programming Gems vol 2", ISBN-10 1584500549, 2001.