사용자 정의 형식을 지원하는 XML 기반 객체 모델의 구조 설계

이용헌*, 심준용*, 김세환*
*LIG넥스원 Maritime연구센터 VM팀
e-mail: yhlee80@lignex1.com

The Design of XML based Object Model Structure supports User-defiend type

Yongheon Lee, Jun-Yong Shim, Sae-Hwan Kim
LIG Nex1 Co. LTD. Maritime R&D Center, VM Team

요 약

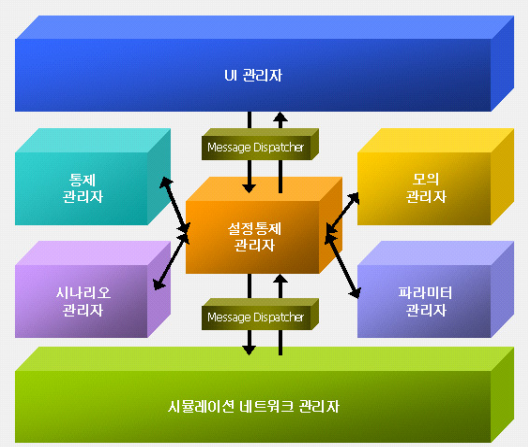
기존의 M&S 프레임워크는 컴포넌트 간에 송수신되는 메시지의 구조가 유연하지 못한 문제점을 가지고 있다. 플러그인 기반 아키텍처는 소프트웨어를 구성하는 컴포넌트들을 플러그인 형태로 구현하여 컴포넌트간의 결합도를 낮추고 유연성 및 재사용성을 강화할 수 있는 구조를 가지고 있다. 이러한 아키텍처를 구성하는 각 컴포넌트는 메시지 지향 미들웨어 기반의 메시지 통신을 수행하게 되는데, 플러그인 간에 종속성이 생기지 않는 형태로 설계되고 구현되어야 한다. XML기반의 객체모델은 이러한 메시지 통신에 사용되는 메시지 객체의 구조를 정의한다. 이 객체모델은 사용자 정의 형식을 지원하며 이러한 형식을 조합하여 새로운 복합 형식을 정의하여 메시지 구조를 표현할 수 있도록 한다. 객체 모델에서는 각 사용자 정의 형식과 각 형식에서 사용하는 기본형(Primitive Type)의 클래스를 추상화하여 정의함으로써 객체 모델의 유연성을 높일 수 있는 구조를 가지고 있다.

1. 서론

최근의 소프트웨어 개발은 그 규모가 점점 거대해지고 복잡해지는 추세에 있고 여러 체계 간의 상호 연동을 매우 중요시하고 있다. 이런 추세에 따라 소프트웨어 개발에 있어서 더 짧은 개발 기간과 더 적은 개발비용으로 개발할 수 있는 환경과 능력이 더욱 요구되고 있다. 이러한 요구를 만족시키기 위하여 소프트웨어의 재사용성(reusability), 유연성(flexibility)을 극대화할 수 있는 개발 환경이 필요하다.

이러한 요구사항을 만족시키는 개발 환경을 구축하기 위하여 기존에 개발되어 사용하고 있는 컴포넌트 연동 프레임워크(M&S 프레임워크)[1][2]는 컴포넌트 재사용 및 컴포넌트 간의 연동이 가능한 형태이지만 컴포넌트 간에 주고받는 메시지의 구조가 유연성이 떨어지는 문제가 존재하여 전체 프레임워크의 유연성을 감소시키는 결과를 초래했다.

이러한 기존 프레임워크의 문제점을 개선하기 위한 개발 환경의 일환으로 플러그인 기반 아키텍처가 도입되었으며 본 연구에서는 이러한 플러그인 기반 아키텍처의 메시지 객체 모델의 구조에 대해 설명한다.



(그림 1) 프레임워크 구성도

2. 관련 연구

2.1 기존의 컴포넌트 연동 아키텍처 & 프레임워크

기존에 당사에서 개발된 컴포넌트 연동 아키텍처 & 프레임워크(M&S 프레임워크)는 프레임워크를 구성하는 각 컴포넌트(관리자) 사이에 Message Dispatcher가 존재하여 메시지를 전달할 수 있는 기본 구조를 지니고 있다. 프레임워크에서는 시뮬레이션 소프트웨어를 구성하는 기본 기능을 각 컴포넌트에 구현하여 제공하고 사용자 정의 컴포넌트를 Message Dispatcher와 결합하여 컴포넌트의 재사

용성과 소프트웨어의 개발 편의성을 높일 수 있도록 하였다. 이러한 구조는 각 컴포넌트의 재사용성을 높일 수 있는 구조이지만, 실제 구현에서는 컴포넌트 사이에 송수신되는 메시지의 명세가 프로그램의 소스코드에서 기술되기 때문에 메시지의 변경이 어렵고 각 컴포넌트가 메시지에 종속이 되는 문제가 발생한다.

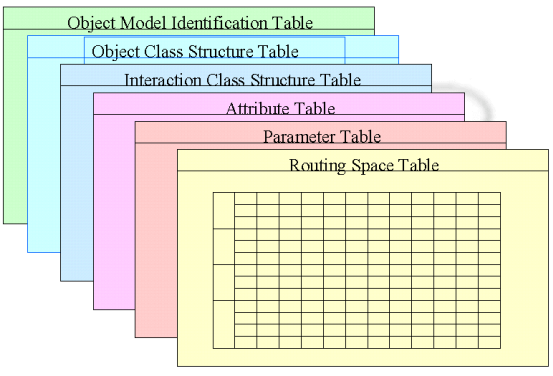
2.2 플러그인 기반 아키텍처[3]

플러그인 기반 아키텍처는 소프트웨어를 구성하는 각 컴포넌트를 플러그인으로 구현하여 Run-time에 이를 조합하여 플러그인의 기능을 수행할 수 있는 기반 구조를 가지고 있다. 플러그인 기반 아키텍처에서는 소프트웨어 실행 중에 플러그인이 교체되거나 새로이 삽입되고, 재컴파일 없이 플러그인의 교체만으로도 소프트웨어의 실행이 가능해야 하므로 플러그인 컴포넌트의 응집도와 유연성이 높아서 컴포넌트 사이의 종속성이 매우 낮아야 한다.

이를 위하여 모든 플러그인 컴포넌트는 하나의 추상화된 클래스를 상속받아 구현되는 구조를 기본적으로 가지게 되며 컴포넌트 간의 데이터 교환 인터페이스 역시 추상화된 메시지 객체를 주고받는 형태를 취하게 된다.

2.3 Object Model Template

Object Model Template(OMT)[4]은 IEEE 표준(HLA, IEEE 1516.2-2000)으로써 HLA 시스템의 각 Federation 및 Federate 들이 사용하는 연동모델(FOM; Federation Object Model, SOM; Simulation Object Model)의 Object/Attribute와 Interaction/Parameter을 정의하는 방법을 규정하고 있다. 이러한 OMT를 준수한 연동모델을 사용함으로써 연동 시스템 사이의 메시지 데이터를 명확하게 정의할 수 있게 된다. 본 연구에서는 OMT의 여러 구성 요소 중에서 Object Class와 Attribute의 구조를 참조하여 객체 모델의 구조 설계에 적용하였다.

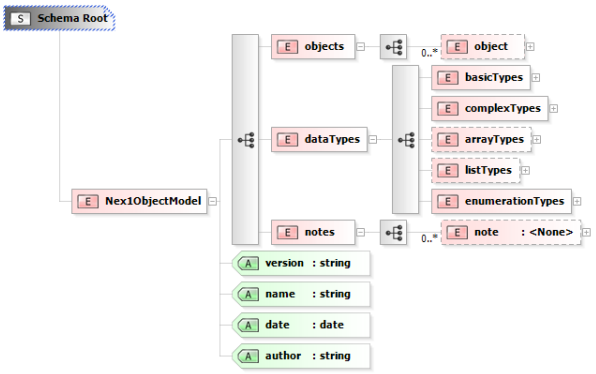


(그림 2) Object Model Template의 구성요소

3. XML 기반 객체 모델

XML(eXtensible Markup Language)는 W3C(World Wide Web Consortium)의 표준으로, 데이터의 구조 등을 기술하기 쉬운 특징을 가지고 있어서 많은 분야에서 사용되고 있다. XML은 플랫폼 독립적(Platform Independent)이며 가독성(Readability)이 높고 이를 지원하는 도구와 라이브러리가 다양하다는 장점을 가지고 있다. 본 연구에서 제시하는 플러그인 기반 아키텍처의 객체 모델은 이러한 XML을 통해서 기술되며 이는 XML 스키마[5]에 의해 구조적으로 정의된다.

객체 모델에 적용된 XML 스키마에서는 플러그인 컴포넌트를 구현하는 개발자가 사용할 각종 기본형(Primitive Type; integer, short, double, etc.)과 열거형(Enumeration Type) 및 이들을 조합한 복합 형식(Complex Type)을 지정할 수 있게 하고, 이러한 타입을 속성(attribute)으로 가지는 메시지 객체(object)를 정의할 수 있는 구조를 가지고 있다. 이 구조는 OMT에서 정의하는 구조를 간략화 시킨 것으로, OMT의 Object Class Structure Table과 Attribute Table을 XML 스키마를 통해 구조적으로 정의할 수 있도록 해주고 있다.



(그림 3) 객체 모델 XML 스키마

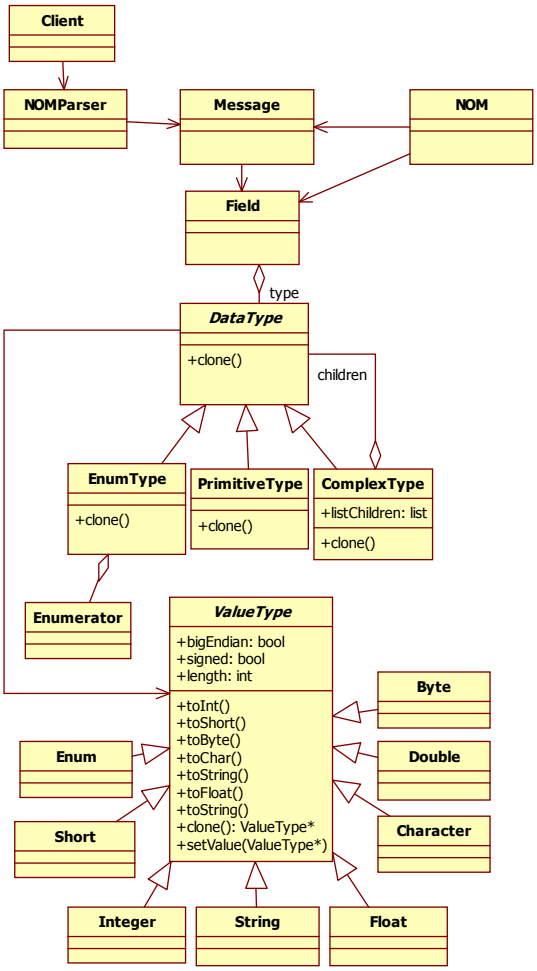
플러그인 컴포넌트 간의 메시지 전송은 컴포넌트의 종속성을 제거하기 위하여 단일 인터페이스를 통해야 하며 전달되는 메시지 객체도 추상화된 형태를 가지고 있어야 한다. 이에 따라 XML 문서에 기술된 객체 모델은 추상화된 메시지 클래스 형태로 변환이 되며 이 클래스의 인스턴스가 플러그인 컴포넌트간에 송수신 된다.

4. 객체 모델 구현

XML 문서로 기술된 사용자 정의 타입과 객체는 실제로 플러그인 컴포넌트간에 송수신되는 메시지로 구현되어야 한다. 이 메시지는 앞서 언급한 바와 같이 하나의 추상화된 클래스로 구현이 되어야 플러그인 아키텍처의 요구사항을 만족시킬 수 있다. 이는 객체 모델에서 사용하는 type, object, attribute의 종류와 수량에 관계없이 단일화된 인터페이스를 제공해야 함을 의미한다. 이를 위하여 그림 3과 같은 클래스 구조를 적용하였다.

XML 문서에 정의된 기본 타입(PrimitiveType class)과 열거 타입(EnumType class)을 조합하여 새로운 복합 타

입(ComplexType class)를 생성하고 이를 반복적으로 수행하는 구조를 표현하기 위하여 GoF[6]의 컴포지트(Composite) 패턴을 적용하였다. 모든 타입은 DataType class로 추상화되며 ComplexType은 DataType class의 인스턴스를 자식객체로 가지는 Recursive한 구조를 가지고 있다. 이러한 구조에서는 PrimitiveType 및 EnumType을 조합하여 새로운 ComplexType을 생성하고 이들을 새로이 조합하여 다른 ComplexType을 반복적으로 무한히 생성할 수 있게 된다. 반대로 생성된 최상위 ComplexType의 자식 객체들의 재귀적인 접근을 통하여 하위의 모든 PrimitiveType과 EnumType을 접근할 수 있게 된다.[7]



(그림 4) 객체 모델의 클래스 다이어그램

이러한 재귀적인 DataType의 구조는 XML문서를 parsing하는 과정을 통해 만들어진다. Parsing이 완료되면 XML문서에 기술된 사용자 정의 타입과 메시지 구조가 테이블 형태로 만들어진다. 실제 컴포넌트간에 메시지를 주고받기 위해서는 메시지 클래스의 인스턴스가 필요하며, 이 인스턴스의 값을 설정하기 위한 ValueType 클래스를 사용하게 된다. ValueType 클래스는 객체 모델에서 제공하는 원시 타입(Integer, Double, etc.)에 대한 Wrapper 클래스로써 역시 추상화된 형태로 표현된다. 이러한 DataType 클래스와 ValueType 클래스는 메시지 클래스의 인스턴스가 생성될 때 인스턴스의 메모리 영역에 복사가 되어야 한다. 단일화된 메모리 복사 인터페이스를 위하여 GoF의 프로토타입(Prototype) 패턴을 DataType 클래스와 ValueType 클래스에 적용하였다. 인스턴스의 메모리 복사를 수행할 때 추상화된 클래스의 clone() 메서드를 호출하면 실제 하위 클래스 객체의 clone()이 호출되어 메모리 복사가 일어나게 된다. 따라서 인스턴스 복사시 메시지 Attribute의 타입과 값에 관계없이 단일화된 인터페이스를 통하여 인스턴스 복사가 이루어질 수 있다.

XML문서 parsing으로부터 메시지 인스턴스 복사의 과정을 거치게 되면 다른 플러그인 컴포넌트로 보내는 메시지의 형태를 갖추게 된다. 여기서 메시지 전송을 위해서 메시지(Object)의 각 필드(Attribute)의 값을 설정하는 과정이 필요하다. 위의 클래스 다이어그램에서 보여지는 것처럼 하나의 DataType은 하나의 ValueType과 매칭이 되며 Message 클래스에서는 이 ValueType의 값을 설정할 수 있는 서비스를 제공한다.(SetValue) 반대로 설정된 값을 얻어올 수 있는 서비스도 함께 제공된다.(GetValue) 이때 XML 문서에 기술한 Attribute의 이름을 통하여 DataType과 ValueType에 접근하게 되는데, 컴파일 시간에 값 변수의 주소가 결정되는 일반적인 구조체 멤버 접근 방식에 비해 실행 중에 동적으로 타입과 값의 객체가 변하는 현재의 구조에서는 값 설정 서비스의 수행 시간이 존재하므로 반복적인 값 설정 서비스 사용은 소프트웨어의 퍼포먼스를 저하시킬 수 있다. 이러한 문제를 해결하기 위하여 메시지의 Attribute 목록에 대해 인덱싱 기법을 적용하여 값 설정 서비스의 수행 시간을 최소로 하는 방법을 적용하였다.

이렇게 실행중에 동적으로 생성된 메시지와 필드의 복합 구조는 플러그인 컴포넌트 사이에 송수신된다. 이를 플러그인 컴포넌트가 아닌 외부 시스템에 송수신하기 위해서는 외부 인터페이스에 적합한 형태로 메시지가 변환이 되어야 한다. 이는 Serialize, Deserialize 서비스를 통해 구현된다. 메시지를 구성하는 기본 단위는 ValueType의 구체 클래스이며 이는 메모리 상에 구현된 자신의 포인터와 데이터 크기를 알고 있다. 이러한 기본단위의 조합을 통하여 메시지를 생성하므로, 각 기본 단위의 메모리 포인터를 조합하여 전체 메시지의 메모리 블록을 생성해낼 수 있다.(Serialize) 반대로 외부 인터페이스로부터 수신된 데이터를 통하여 메시지 데이터를 생성할 수도 있다.(Deserialize) 기존 프레임워크에서는 외부 인터페이스와 연결하기 위한 데이터 변환을 개발자가 직접 메시지의 필드별로 코딩해야 하는 문제점이 존재했다. 본 연구의 객체 모델에서는 그림 5와 같이 다양한 서비스를 제공함으로써 개발자의 개발편의성을 높이고 있다.

이러한 객체 모델의 구조는 플러그인 컴포넌트 사이에 송수신 될 수 있는 단일 인터페이스를 제공하면서도 실행 중에 동적으로 데이터 구조가 생성될 수 있는 유연성을

갖추고 있다. 이와 함께 개발자의 개발 편의성을 높일 수 있는 서비스들을 제공함으로써 기존 개발된 프레임워크를 대체하는 유연성 있는 프레임워크를 개발할 수 있다.

(그림 5) 객체모델 제공 서비스

5. 결론 및 추후 연구 방향

본 연구에서 제시한 객체 모델은 기존 개발된 프레임워크 컴포넌트 사이의 종속성 문제를 해결할 수 있다. 플러그인으로 구현된 각 컴포넌트는 XML로 기술된 사용자 정의 형식의 메시지 목록을 사용하여 인터페이스의 변경 없이 컴포넌트간 메시지 송수신이 가능하다. 이와 더불어, 객체 모델에서 제공하는 서비스들을 통하여 유연하고 편의성 높은 개발환경을 구축할 수 있다.

위의 객체 모델은 GoF의 디자인 패턴과 같은 객체 지향 기법이 적용되어 있다. 이러한 객체 지향 기법은 동적인 데이터 구조와 객체의 동적 바인딩(Dynamic Binding)에 의하여 수행되므로 컴파일 시간에 모든 것이 결정되는 정적 바인딩 기법에 비해 속도가 느린 단점을 가지고 있다. 이를 위하여 추후에는 객체 모델의 성능 향상을 중점으로 연구를 진행해 나갈 것이다.

참고문헌

[1] 이용헌, 이승영, 정하민, 김세환, “분산시뮬레이션 환경의 운용통제 프레임워크 설계”, 소프트웨어공학 합동워크샵, 2008.07

[2] 심준용, “M&S Framework를 적용한 효율적인 분산객체 통신모듈 설계”, 한국소프트웨어공학 학술대회 논문집 제10권 1호, 2008

[3] 원강연, 최상영, “M&S PlugIn-Based Architecture Framework 개발”, 한국정보과학회논문지, 시스템 및 이론 제36권 제2호(2009.4)

[4] http://www.msco.mil/HLAComplianceTesting.html

[5] http://www.w3.org/XML/Schema

[6] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software", Addision-Wesley, 1994

[7] 이용헌, 심준용, 조규태, 이승영, 김세환, “Publish-Subscribe 메시지 프로토콜을 위한 XML 기반의 Object Model 설계", 한국시뮬레이션학회 춘계학술대회, 2010.05