Model Driven Architecture 기술 소개

An Introduction to the Model Driven Architecture

김우식(W.S. Kim) 컴포넌트공학연구팀 연구원
권오천(O.C. Kwon) 컴포넌트공학연구팀 책임연구원
신규상(K.S. Shin) 컴포넌트공학연구팀 책임연구원, 팀장

2001년 9월 OMG는 시스템을 명세하고 구축하기 위한 새로운 방안으로 MDA(Model Driven Architecture)를 표준으로 채택하였다. MDA는 OMG가 지금까지 추진한 여러 표준화 작업을 바탕으로 한 모델 중심의 시스템 명세와 개발에 대한 것이다. OMG는 그 동안 CORBA를 통하여 구현 수준에서 기술간 상호 운영성과 시스템 통합을 다루어 왔다. OMG는 여기서 더 나아가 UML, MOF, CWM 등의 모델링 표준화 작업의 결과를 기반으로 한 MDA로 구현 수준에서 설계 수준으로 시스템 통합과 기술간 상호 운영성 문제 해결을 시도한다. 본 고에서는 MDA에 대한 개괄적 설명과 함께 MDA의 기반을 이루는 관련 표준에 대해서 알아보고 그 현재 상황을 살펴본다.

I. 서 론

OMG(Object Management Group)는 1989년에 최초로 설립된 이후부터 지금까지 계속해서 시스템 통합(integration)과 기술 간의 상호 운영성(interoperability)에 대한 일을 추진해 왔다. 그 노력의 일환으로 벤더 중립적이고 언어 독립적이며 다른 미들웨어를 포함할 수 있는 미들웨어인 CORBA(Common Object Request Broker Architecture)를 만들어 냈다. CORBA 프로젝트는 성공적으로 수행이 되어 지금은 real-time system, embedded system, fault-tolerant system 등의 많은 분야로 그 응용 영역을 확대하고 있다.

OMG의 CORBA는 컴퓨팅 분야에서 괄목할 만한 성과를 거두었지만 지금의 기술적 상황에서 여전히 시스템 통합과 기술간 상호 운영성 문제는 지속적으로 해결되어야 할 과제로 남아 있다. 이것에 대한 여러 가지 이유가 있겠지만 두 가지 정도의 주요한 이유가 있다.

우선, IDL(Interface Description Language)을 이용한 단일 인터페이스 정의를 통해 얻을 수 있는 상호 운영성은 제한적이기 때문이다[1]. 이미 소멸된 언어로 작성된 레거시 시스템(legacy system) 모두를 IDL 표준에 맞게 작성하는 것은 가능하지 않다. 또한 웹의 확산과 더불어 고유의 인터페이스를 가진 비즈니스 파트너와의 연결이 지속적으로 요구되는 상황에서 시스템 개발자는 서로 다른 컴포넌트를 연결하기 위해서 인터페이스를 조정하는 소비적 노력을 계속해야 한다.

다음으로는 다양한 미들웨어의 확산 때문이다[2]. 물론 CORBA가 시장에서 충분히 성공적이었지만 여전히 수많은 기업들은 EJB, XML/SOAP, COM+, .NET 등 여러 가지 미들웨어를 사용하고 있다. 더욱

주) 본 연구는 2002년 과학기술부 국가지정연구실 사업으로 수행되었음

이 문제가 되는 것은 기업 수준의 시스템인 경우 내부 조직의 서로 다른 요구 사항 때문에 하나의 미들웨어만을 가지고 기업 전체의 시스템을 구성하는 것은 쉽지 않다. 또한 하나의 미들웨어를 통한 기업 전체의 시스템 구성이 가능한 경우라도 B2B 시장의 확대로 다른 기술기반을 가진 외부 기업과의 통합은 피할 수 없다. 다시 말하면 기술 상황이 지속적으로 변화하기 때문에 특정 기술에 기반한 시스템을 기술 변화에 맞게 통합, 변화, 유지하는 것은 여전히 문제로 남아 있다. 이러한 문제점을 해결을 위해서 OMG는 MDA(Model Driven Architecture)[1]라는 개발 패러다임을 제시하였다.

MDA는 OMG가 지금까지 지속적으로 추구해 왔던 시스템 통합의 연장선에서 2000년 11월 OMG에서 제안되었다.[2]

MDA는 시스템의 설계와 명세를 정형화된 모델로 기술 플랫폼과 분리하여 기술하고, 실제 구현과 관련된 모델은 매핑을 통해서 기술 플랫폼에 독립적으로 기술된 모델을 변환하여 얻는 방법에 대한 것이다. 이 두 모델은 MDA에서는 PIM(Platform Independent Model)과 PSM(Platform Specific Model)이라고 한다. PIM과 PSM은 모두 UML(Unified Modeling Language)로 기술된다.

MDA 접근 방법은 시스템을 PIM을 통하여 특정 기술에 특화된 PSM을 만들어냄으로써 기반 기술이 변화하더라도 PIM 변환을 통해서 해당 기술변화에 대응하는 PSM을 만들어 냄으로써 시스템을 보다 효율적으로 유지할 수 있다. 이렇게 하나의 시스템을 PIM과 PSM으로 기술하는 것은 좀더 유연하고 생산성 높은 모델 수준의 시스템 통합을 이루게 할 수 있다.

II. MDA

1. MDA의 개요

MDA는 CORBA와 같이 실제 구현 수준의 표준이 아닌 설계 수준의 표준이다[3]. MDA는 UML을 통해서 시스템을 모델의 형태로 설계 및 명세하는 것과 시스템의 개발 라이프 사이클 전체에서의 모델 역할에 대한 것이다.

MDA 개발 방법은 우선 PIM을 정의함으로 시작한다. PIM은 특정한 기술 플랫폼이나 기반 기술에 독립적인 방법으로 시스템을 설계한 모델을 말한다. PIM은 단일 플랫폼에 매여 있지 않고 프로그래밍 언어 독립적이기 때문에 다른 시스템으로의 가교 역할을 할 수 있다. MDA에서는 이러한 PIM에 정형화된 변환 법칙을 사용해서 PSM을 생성한다. PSM은 특정 기술에 종속적인 요구사항들이 포함된 시스템 모델 정보이다. 여기서 PIM과 PSM 모두 UML로 기술된다. 마지막으로 PSM을 통해서 실제로 동작하는 구현 코드를 만들어 낸다. (그림 1)은 이와 같은 방법으로 단일한 PIM을 통해서 다중 플랫폼으로 구현이 이루어지는 경로를 보여준다.

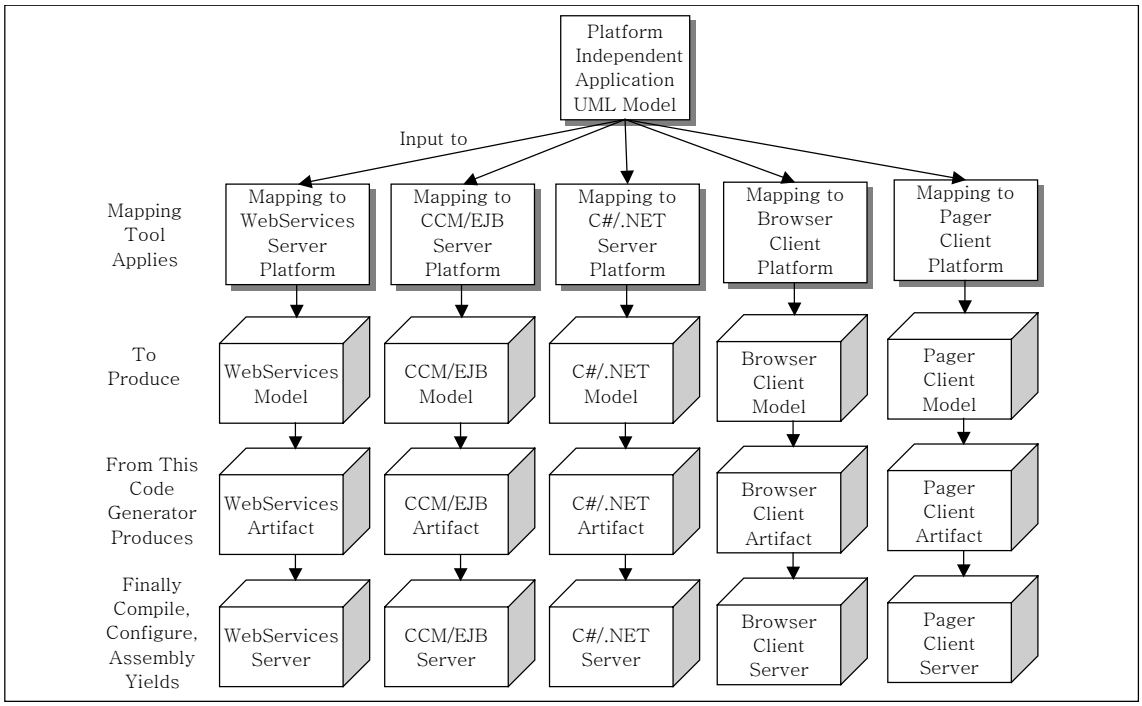
MDA 방식으로 개발된 시스템은 다음과 같은 이점을 지니게 된다[3].

- 기술 변화 상황에 효율적으로 대처할 수 있다. MDA 방식으로 개발된 시스템은 PIM을 통해서 변경된 기술 플랫폼으로 이식이 쉽게 이루어질 수 있기 때문이다.
- 시스템 인프라 변화에 유연하게 대처할 수 있다.
- MDA 방식으로 개발된 시스템은 그 시스템의 유지 보수 비용이 적으며 시스템의 수명이 길다. 따라서 투자 비용이 보존된다.

이러한 이점은 PIM과 PSM으로 시스템의 모델 정보를 기술 종속적인 부분과 기술 독립적인 부분으로 분리하고 단일 PIM으로부터 특정 기술에 대한 PSM을 자동화된 기법으로 생성함으로 얻어지는 것이다. 이러한 개발 방식의 변화는 개발 과정에서 시스템 설계와 분석 및 코드 생성의 도구로서의 모델의 위치를 시스템의 유지 보수, 전개 등을 포함하는 시스템 개발 라이프 사이클 전체로 확장한다.

이러한 MDA는 OMG가 그 동안 추진한 모델링 표준화 작업이 어느 정도 성숙되었기 때문에 가능한

1) "모델 구동형 아키텍처"라고도 한다.
2) 2001년 9월에 표준으로 채택되었다.



(그림 1) 단일 PIM을 이용한 멀티 플랫폼 개발

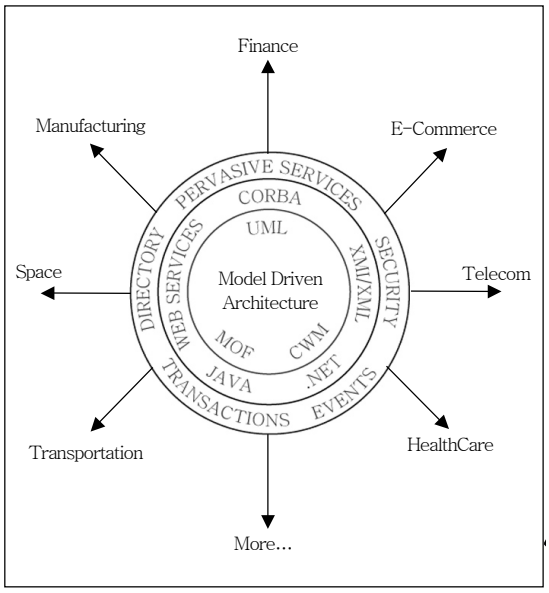
것이다. MDA는 모델이 개발 중심이기 때문에 모델 정보에 대한 표준화된 기술, 접근, 교환, 보관 등의 방법이 필요하다. 이러한 요구 사항들은 아래의 OMG 표준에 의해서 강력하게 지원 받는다.

- UML(Unified Modeling Language): 객체 및 컴포넌트 시스템을 표현하기 위한 표준언어이다.
- MOF(Meta Object Facility): 모델 정보에 대한 표준적인 저장소를 제공하고 표준화된 방식으로 모델 정보를 접근하는 구조를 정의한다.
- CWM(Common Warehouse Metamodel): 데이터 저장소 통합에 대한 표준을 정의하고 데이터베이스 모델과 스키마 변환 모델, OLAP, 데이터 마이닝 모델(data mining model)에 대한 표준화된 표현 방법을 제공한다.
- XMI(XML Metadata Interchange): UML로 기술된 모델 정보의 XML 표현에 대한 표준이다.

지금까지의 MDA에 대한 설명은 (그림 2)에 잘 설명되어 있다. 그림의 중심에 위에서 언급한 OMG의 모델 관련 표준인 UML, CWM, MOF가 자리 잡고 있는 것은 MDA가 모델 중심임을 나타낸다. 그리고 중심에서 밖으로 두번째 원은 코어 모델을 통해서 생성해 낼 수 있는 대상 기술 플랫폼을 나타낸다.

시스템을 PIM으로 설계하기 위해서 추가적으로 필요한 작업 중의 하나는 기본 서비스(pervasive service) 즉, 디렉토리(directory), 보안(security), 분산 이벤트 처리(distributed event handling), 트랜잭션(transaction), 지속성(persistence) 서비스 등을 플랫폼 독립적으로 설계하는 작업이다. 하나의 시스템 또는 애플리케이션에서 사용하는 기본 서비스가 PIM 수준에서 모델링 된다면 이를 통해서 PIM으로 기술된 기본 서비스를 지원하는 모든 미들웨어에 손쉽게 해당 시스템이나 애플리케이션이 구현될 수 있다. 현재 OMG에서는 그간 CORBA에 종속적으로 기술된 기본 서비스를 PIM 방식으로 기술하는 작업을 진행중이다.

마지막으로 (그림 2)에서 화살표로 표현된 것은



(그림 2) OMG의 MDA

재무, 전자거래, 통신, 의료, 교통, 우주항공 제조 등 특정 도메인의 공통적인 비즈니스 요구 사항을 PIM 모델로 기술함을 나타낸다. 이는 PIM으로 기본 서비스를 기술한 것과 같은 맥락으로 생각할 수 있다.

이처럼 MDA는 비즈니스 모델을 IT 하부구조(infrastructure)와 분리함으로써 기술 변화 상황에 적절하게 대응하고 시스템 통합을 용의하게 함으로 생산성을 얻는 것을 목표로 한다.

2. MDA의 구조

MDA는 핵심이 되는 시스템 모델들을 분리하고 그들 사이에 일관된 구조를 정의한다.

MDA는 한 시스템의 PIM과 PSM의 구조와 관계를 정의할 뿐만 아니라 연관 관계를 가지는 시스템간의 PIM과 PIM 또는 PSM과 PSM의 관계에 대한 정의도 포함한다. (그림 3)은 3개의 추상화 수준으로 모델링된 두 개의 시스템과 두 시스템의 같은 추상화 수준에 있는 모델간의 연관관계를 보여 주는 MDA의 모델간 관계의 정형적인 구조이다.

우선 두 시스템이 각각 기본적으로 PIM과 PSM으로 분류되어 있다. 그리고 PIM을 좀더 추상화 함으로써 얻어지는 비즈니스 모델이 포함된다. 이 비즈니스 모델이 포함하는 것은 PIM에서 나타날 수 있는 컴퓨팅 환경에서의 수행을 가정한 여러 가지 요소들을 제외하고 비즈니스 요구 사항만이 포함된 모델 정보이다.

이렇게 모델을 PIM과 PSM으로 분리하는 것은 아래와 같은 이점을 지니고 있기 때문이다[5].

- 플랫폼에 종속적인 사항을 고려하지 않아도 되므로 PIM 수준에서의 모델의 완전을 확인하기가 쉽다.
- PIM을 통한 PSM의 생성을 통해서 기술변화에 유연하게 대처할 수 있다.
- 시스템의 통합과 상호 운영성을 PIM 수준에서 정의함으로 좀더 유연한 방식으로 해결될 수 있다.

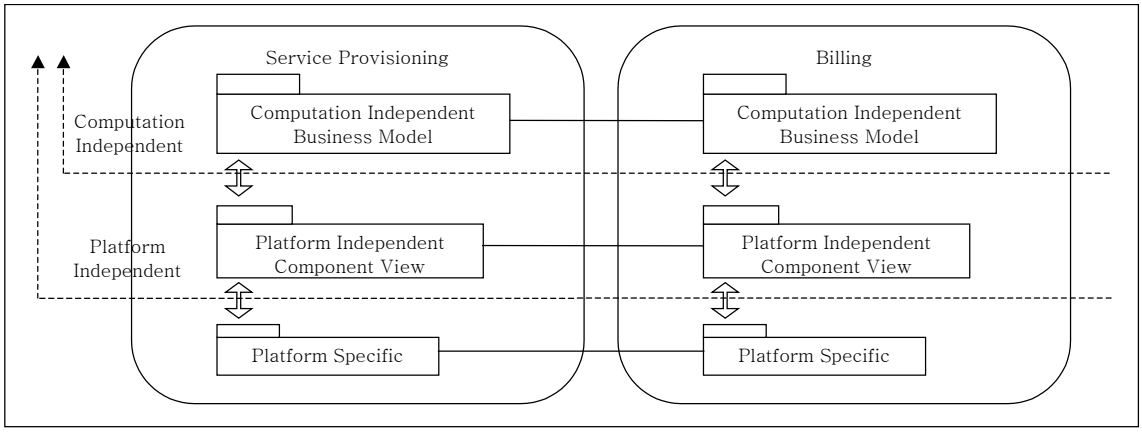
그리고 MDA에서는 각 모델간의 일관된 관계를 정의한다. 우선, 하나의 시스템 내에서 비즈니스 모델, PIM, PSM으로 이어지는 모델간의 관계를 정의한다. 그리고 타 시스템과 해당 시스템에서 같은 추상화 수준을 가지는 모델간에, 즉 비즈니스 모델과 비즈니스 모델, PIM과 PIM, PSM과 PSM 사이의 관계를 정의하는 것을 포함한다.

이러한 모델간 구조에서 PIM과 PSM의 분리와 더불어 중요한 점은 PIM에서 PSM으로의 변환과 PSM에서 실제 코드로의 변환이다. 특히 이러한 변환이 자동화 된다면 MDA를 통해서 얻을 수 있는 생산성은 매우 향상될 수 있을 것이다. MDA에서는 PIM to CORBA나 PIM to EJB와 같이 PIM에서 특정 PSM으로의 정형화된 변환 규칙과 패턴이 존재하며 이를 통해서 높은 수준의 변환의 자동화가 가능함을 말하고 있다.

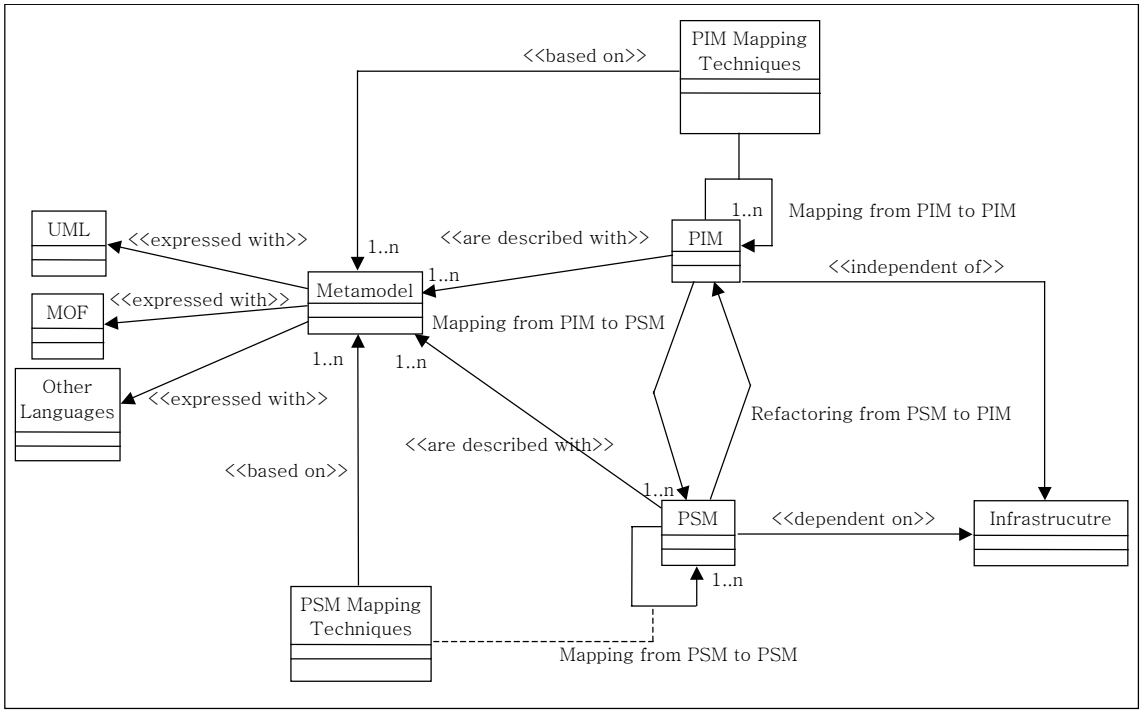
3. MDA의 모델 변환

(그림 4)는 PIM과 PSM 그리고 그들간의 매핑 방법을 OMG의 핵심 표준들인 MOF, CWM, UML로 기술되는 메타모델을 기반으로 설명하고 있다.

MDA에서의 매핑은 하나의 모델에서 다른 모델



(그림 3) MDA에서의 모델 분리와 모델간 관계



(그림 4) MDA 메타모델로 설명되는 PIM과 PSM의 모델 변환

<표 1> 모델 변환의 종류

PIM to PIM	PIM에서 PIM으로의 변환은 PIM이 개발 단계에서 좀더 상세화되는 경우다. 상세화가 이루어지더라도 변환된 PIM에 특정 기술관련 사항이 기술되지 않음
PIM to PSM	PIM이 실제 시스템의 기능을 충분하게 기술하고 있다면 기술 종속적인 정보를 추가하여 PSM으로 변환한다. 변환된 PSM도 UML로 기술함
PSM to PSM	PSM과 PSM의 변환은 PIM to PIM 관계와 같이 상세화 관계이다. 하지만 이 변환에서는 실제 구현과 관련된 정보가 포함된다. 예를 들면 컴포넌트가 실제로 실행되기 위한 컨테이너 정보, 설정 정보, 배포 정보를 추가하는 것이 이러한 변환에 해당함
PSM to PIM	기존의 시스템의 구현 상황을 추상화하여 PIM 모델을 얻어내는 과정임

로 변환하기 위한 일련의 규칙들의 집합을 말한다. <표 1>은 MDA에서 가능한 모델 변화의 종류를 설명하고 있다[1].

4. UML Profile

UML Profile은 UML의 기본 빌딩 블록들을 특별한 목적에 맞도록 UML의 stereotype과 tagged value 두 가지 자체 확장 메커니즘을 사용해서 확장한 것이다.

MDA에서 UML 프로파일은 중요한 위치를 점하고 있다. MDA에서 PSM은 UML로 표현되어야 한다. 하지만 특정 기술 플랫폼의 개념들이 UML로 어떤 식으로 표현되어야 하는지에 대한 것은 UML 자체에 포함되어 있지 않다. 예를 들면 EJB의 경우 entity bean, session bean, deploy descriptor의 내용들을 표준화된 방식으로 UML로 표현할 필요가 있다. 이를 위해서 OMG에서는 해당 플랫폼에 따라서 UML 프로파일을 만드는 작업을 하고 있다. 현재 UML Profile for CORBA의 스펙 정의는 종료된 상태이며, JCP(Java Community Press)에서 EJB 1.1을 기반으로 UML Profile for EJB 1.0 draft가 나와 있다[4]. UML Profile은 UML Profile for .NET 또는 UML Profile for SOAP와 같이 현재 널리 쓰이는 기술 플랫폼에 대해서 만들어질 것이며, 앞으로도 새로운 기술 플랫폼이 등장할 때마다 OMG에 의해서 만들어 질 것이다.

MDA에서 UML 프로파일은 해당 기술 플랫폼을 표현하기 위한 PSM 수준의 도구로만 사용되는 것은 아니다. MDA는 PIM 수준에서 특정 도메인에 일반적인 요소들, 예를 들면 컴포넌트 아키텍처에서의 이벤트, 프로세싱, 엔티티, 패턴 등을 표준화된 방법으로 표현할 필요가 있다. 이러한 목적으로 분산 컴퓨팅 환경에 대해서 UML Profile for EDOC(Enterprise Distributed Object Computing)을 OMG에서 제안하였다. 이 또한 도메인 영역이 늘어감에 따라 UML 프로파일이 늘어날 것이다. 현재 OMG에서 작업하고 있는 UML 프로파일은 <표 2>와 같다.

<표 2> 현재 발표된 UML Profile

UML Profile for CORBA	CORBA를 PSM에서 UML로 표현하기 위한 Profile
UML Profile for EJB	EJB를 PSM에서 UML로 표현하기 위한 Profile
UML Profile for EDOC	Enterprise Application을 PIM 수준에서 기술하기 위하여 Process, Event, Pattern, Enterprise Collaboration Architecture 등의 UML 표현 규칙을 정한 Profile (현재 웹서비스에 대한 PSM 매핑이 작업중에 있다)
UML Profile for EAI	Loosely-coupling System을 위한 PIM 수준의 Profile
UML Profile for Schedulability, Performance, and Time	Real-time System의 Schedulability, Performance 등을 모델 수준에서 정량적으로 분석 가능하도록 표현하기 위한 Profile

III. MDA 관련 OMG 표준

1. UML

UML은 여러 가지 객체지향 방법론들이 통합하여 만들어져서 OMG에 의해 표준화된 객체지향 시스템과 컴포넌트 기반 시스템의 분석 및 설계에 현재 가장 널리 쓰이는 노테이션(notation) 표준이다. UML은 1.3버전에 UML 프로파일 작성에 관한 가이드라인이 추가되어 1.4버전이 정식으로 2000년 말에 발표되었으며, 현재는 UML 2.0에 대해서 작업이 이루어지고 있다. UML 2.0을 위한 주요한 RFP는 다음과 같다[5].

- 기초적인 UML 구조(construct)를 재구성하고, 커스터마이징과 관련한 기능을 발전시키는 것과 관련된 하부구조 RFP
- 컴포넌트, 액티비티(activity), 상호작용 등과 관련된 더 진보한 핵심 패키지를 위한 수퍼구조(superstructure) RFP
- 기본 UML OCL(Object Constraint Language)의 정확성과 표현력을 증진시키기 위한 OCL RFP
- 툴들 사이의 모델 다이어그램을 교환할 수 있도록 하는 다이어그램 교환 RFP

UML 2.0은 여러 가지 면에서 MDA를 지원하고

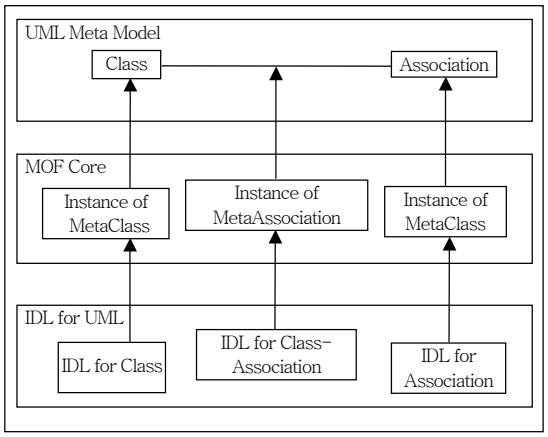
있다. 기본적으로 MDA의 목표인 모델로부터 시스템과 애플리케이션의 자동생성과 관련하여 UML 2.0에서 OCL이 강화되고, 액션 의미(action semantic)가 추가된다. 또한 MDA의 zoom-in, zoom-out과 UML 관련하여 2.0에는 refinement relation에 대한 내용의 추가가 논의되고 있다.

2. MOF

개발과정 전체에서 메타 모델 또는 모델 정보를 정형화된 방식으로 접근하고 유지한다면 생산성과 효율성 면에서 여러 가지 이점을 얻을 수 있다. 하지만 메타 모델과 모델의 다양성 때문에 통일적인 방식으로 모든 모델 정보를 다루는 것은 어려움이 많다. 이러한 문제를 해결하기 위해서 OMG에서 1997년에 MOF를 표준으로 채택했다.

MOF는 메타 모델 또는 모델의 공통 기반 모델이다. 다시 말하면 MOF는 다른 메타 모델을 정의하기 위한 모델이다. 따라서 MOF는 메타-메타모델이라고 할 수 있다. 어떠한 메타 모델이 MOF를 기반으로 만들어졌다면 그 메타 모델은 MOF 기반의 저장소에 저장될 수 있다. OMG의 다른 핵심 표준인 UML과 CWM은 모두 MOF 기반의 메타 모델이기 때문에 같은 저장소에 저장될 수 있다[6].

MOF는 모델 정의를 위한 기본 구성 요소와 구성 요소를 IDL로 매핑하는 MOF-IDL이 포함된 표준이다. 현재 MOF_IDL 매핑은 CORBA IDL로 행해지고 있다. (그림 5)는 MOF의 동작 방식을 보여 준다[7]. UML 클래스는 우선 MOF의 클래스로 표현된다. 그러면 MOF-IDL 매핑을 통해서 UML 클래스에 대한 IDL이 만들어지게 된다. 다시 말하면 UML 메타 모델의 클래스와 어소시에이션에 대한 CORBA-IDL이 만들어지게 된다.



(그림 5) MOF-IDL 매핑과 UML 메타모델 IDL의 관계

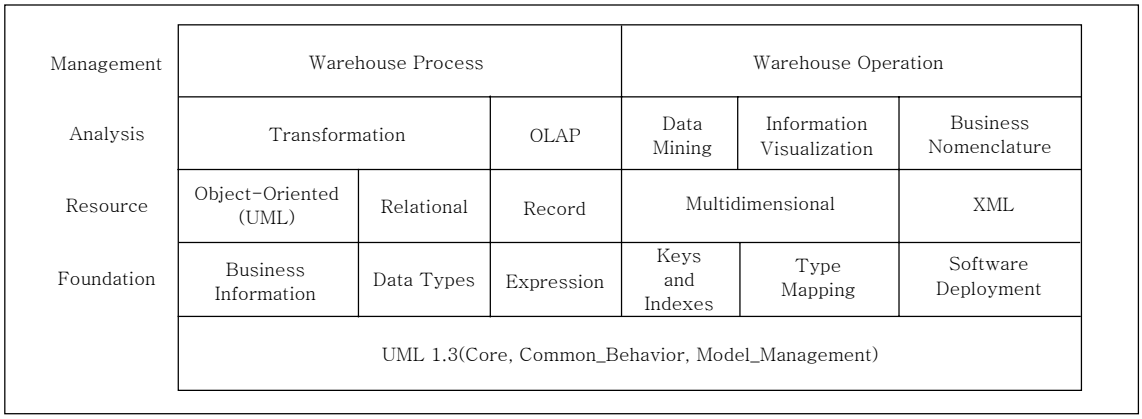
MOF는 MDA에서 모델 저장소의 역할을 수행한다. MOF는 2002년 1월 11일에 1.4버전이 나왔으며 MOF와 UML의 현재의 호환성은 90% 수준이다[9].

3. CWM

CWM은 UML이 애플리케이션 설계 시에 모델링의 도구로서 사용되는 것처럼 데이터 웨어하우징(data warehousing) 영역에서 일반적인 데이터 웨어하우스 아키텍처를 정의한 메타 모델이다. 데이터의 변환은 데이터 웨어하우징의 핵심적인 요소이다. 하지만 모델 변환을 위한 데이터 웨어하우징 저장소 간의 표준화된 교환 규칙과 데이터 모델이 없기 때문에 변환 작업은 쉽게 이루어지기 힘들다. CWM은 이러한 문제를 해결하기 위해서 데이터 소스, 데이터 타깃 그리고 데이터의 변환에 대한 표준화된 모델을 제시한다[8].

CWM은 (그림 6)과 같이 4계층과 18개의 패키지로 구성된다[9].

- Foundation Layer: 나머지 패키지에 대한 공통 서비스를 포함한다.
- Resource Layer: Object-oriented, Relational, Record, Multidimensional, XML 데이터에 대한 메타 모델을 포함한다.
- Data Analysis Layer: 데이터 변환, OLAP(On-Line Analytical Processing), Data Mining, Information Visualization, Business Nomenclature에 대한 메타 모델을 포함한다.
- Management Layer: 웨어하우스 오퍼레이션의 결과와 웨어하우스 오퍼레이션에 대한 메타 모



(그림 6) CWM 패키지 구조

델을 포함한다.

MDA에서 CWM은 데이터베이스 스키마에 대한 MDA 매핑을 위한 역할을 수행한다. CWM은 2001년 2월 1.0버전이 나온 후 현재 1.2 RFP가 나와 있는 상태이다.

IV. 결 론

지금까지 OMG의 MDA에 대한 개괄적인 소개와 더불어 관련 표준에 대해서 알아 보았다. MDA는 특정 기술에 묶여져 있던 개념적 설계를 분리함으로써 기술 여건의 변화에 따라 동일한 개념적 설계를 반복하는 불필요한 작업을 수행하는 노력을 줄여주었다. 또한 설계 수준의 시스템 통합을 기술함으로 지속적인 기술 여건 및 기반 기술의 변화에도 적은 노력으로 시스템의 통합을 이루고 상호 운영성을 유지할 수 있다.

이러한 이점에도 불구하고 OMG의 MDA가 성공적으로 개발자 사회에 받아 들여지기 위해서는 몇 가지 문제점을 해결해야 한다.

지속적으로 변화하는 기술 상황에 따라 수많은 UML 프로파일을 OMG에서 제때 표준화해야 한다. 현재 UML Profile for EJB는 EJB 1.1을 기반으로 만들어 졌지만 표준이 완성되기 전에 EJB 2.0이 발표되었다.

레거시 시스템에 대한 PIM 변환에 대한 것이 좀 더 명확하게 다루어져야 한다. 현재 대부분의 시스템은 빈 공백상태에서 만들어지지 않는다. 새로운 시스템을 개발할 때 기존 클래스나 컴포넌트를 재사용하지 않을 수 없다. 따라서 새로운 시스템이 MDA 방식으로 개발되기 위해서는 기존 시스템을 PIM로 변환하거나 래핑하는 작업이 필요하다.

OMG 표준이 좀더 개발자들이 이해하기 쉽게 되어야 한다. MDA 방식은 설계 수준부터 표준을 따라야 한다는 것이다. 또한 PIM과 PSM의 경계의 구분은 기존 프로그래밍 언어 중심으로 문제를 풀어온 개발자들에게 쉽지 않은 일이다. 따라서 MDA의 기반 표준인 MOF, CWM, UML, XMI 등 개발자들이 좀더 알기 쉽게 단순화되거나 경량화되어야 한다.

MDA는 아직은 시작 단계에 있기 때문에 이상의 문제점들 외에도 여러 가지 문제점이 존재한다. 하지만 MDA가 목표로 하는 설계와 구현의 완벽한 분리와 설계로부터 자동화된 방식의 구현 생성은 컴퓨팅 커뮤니티가 지속적으로 원해 온 것이며, 시스템 개발 방법의 올바른 방향임에는 틀림없다. 따라서 OMG의 지속적인 표준화 작업이 뒷받침되고 현재 이슈가 되고 있는 웹 서비스와 같은 신기술들이 MDA 방식으로 성공적으로 수행된 사례가 늘어난다면 UML과 같이 개발자 사회의 보편적인 개발패러다임으로 머지 않은 시간에 자리잡을 수 있을 것이다.

참 고 문 헌

[1] Object Management Group, "Model Driven Architecture," OMG document ormsc/01-07-01.

[2] Richard Soley and the OMG Staff Strategy Group, "Model Driven Architecture," ftp://ftp.omg.org/pub/docs/omg/00-11-05.pdf, Nov. 2001.

[3] Jon Siegel and the OMG Staff Strategy Group, "Developing in OMG's Model Driven Architecture," ftp://ftp.omg.org/pub/docs/omg/01-12-01.pdf, Nov. 2001.

[4] JSR-000026 UML/EJB™ Mapping Specification 1.0-draft, http://www.jcp.org/aboutJava/community/process/review/jsr026.

[5] 정지훈, "미리보는 UML 2.0," 마이크로소프트, 2002. 5.

[6] Object Management Group, "Meta Object Facility(MOF) Specification, Version 1.4," OMG document formal/2002-04-03.

[7] David S. Frankel, "Business Objects The OMG Meta Object Facility," http://www.adtmag.com/java/articleold. asp?id=1246&mon=3&yr=1999.

[8] Olaf Kath, "Impa cts Of Changes In Enterprises Software Construction For Telecommunications-Model Driven Architecture - Assessments of Relevant Technologies," Jan. 2002.

[9] Object Management Group, "Common Warehouse Metamodel(CWM) Specification, Version 1.0," OMG document ad/01-02-0.