정보산업 동향

MDA 기반 소프트웨어 컴포넌트 아키텍쳐

최성운*, 장진호**, 전인걸***, 김길조****, 홍선주*****

● 목 차 ●

1. 서 론

소프트웨어 컴포넌트(Component : 부품) 기술의 등장은 소프트웨어 산업의 생산성 향상에 새로운 가능성을 제시하고 있다. 이미 제작된 컴포넌트를 조립하여 소프트웨어 시스템을 개발할 수 있다면 획기적인 생산성 및 품질의 향상을 기대할 수 있기 때문이다. 하지만 실제로 컴포넌트 기술을 소프트웨어 개발에 적용하는데 교육, 경험 등 많은 문제점이 발생하고 있으며, 컴포넌트 기술의 이해 부족으로 생산성의 향상을 가져오지 못하는 경우도 종종 발생하고 있다. 이러한 이유로 항상 새로운 기술이 그러하였듯이 컴포넌트 기술 도입 자체가 의문 시 되고 있다.

하지만 컴포넌트 기술이 소프트웨어 생산성 및 품질 향상을 가져올 수 있다는 데에는 이론의 여지가 있을 수 없다. 왜냐하면 소프트웨어 시스템과 하드웨어 시스템의 개발 및 생산 공정은 유사한 구조를 가지고 있으며, 하드웨어 산업의 오랜 역사를 통해, 컴포넌트 기술은 이미 검증되었기 때문이다. 다만 소프트웨어 산업에 컴포넌트 기술을 도입하기 위해서는 소프트웨어의 특성을 고려한 보다 체계적이고 조직적인 방법이 요구되고 있다.

* 명지대학교 컴퓨터공학부 부교수
** 한국전자통신연구원 책임연구원
*** 한국전자통신연구원 연구원
**** 한국전자통신연구원 선임연구원
***** 명지대학교 컴퓨터공학과 대학원 재학

2. S/W 컴포넌트 기술

2.1 소프트웨어 시스템의 특성

컴포넌트를 조립함으로써 소프트웨어 생산성의 향상을 기대할 수 있다는 가정은 소프트웨어 및 하드웨어 시스템 개발 시 동일하게 적용될 수 있다. 하지만 소프트웨어 시스템의 경우 하드웨어 시스템의 조립 생산과는 차이를 가진다. 하드웨어 시스템의 조립 생산의 문제는 동일한 시스템을 대량 생산하는 단계에 집중되어 있는 반면, 복제가 용이한 소프트웨어 시스템의 경우에는 그 초점이 새로운 시스템을 개발하는 단계에 집중되어 있기 때문이다. 즉 개발된 소프트웨어 시스템의 대량 생산은 소프트웨어 복제를 통해 쉽게 가능하다. 소프트웨어 조립 생산의 문제는 대량생산의 문제가 아니라, 특화된 요구에 적합한 다양한 시스템들의 개발 문제인 것이다.

2.2 컴포넌트 기술의 본질

소프트웨어 컴포넌트 기술은 다양한 시스템에 개별적으로 적용되어야 하는 기술이다. 이러한 이유로 컴포넌트 기술은 서로 다른 기종의 컴퓨터 및 운영체제에 공통적으로 적용될 수 있어야 한다. 또한 컴포넌트 자체의 개발 보다는 컴포넌트를 조립하기 위한 시스템의 구조(Architecture)개발에 초점을 맞추어야 한다. 컴포넌트 구조는 컴포넌트가 조립되어야 할 틀이며, 이 구조 속에서 컴포넌트의 역할 및 인터페이스가 정의되기 때문이다. 컴포넌트 기술은 컴포넌트를 제작하는 기술이 아니라 시스템의 구조에 맞추어 관련 도구들을 이용하여 컴포넌트를 조립하는 기술이다.

2.3 컴포넌트 기술의 방향

컴포넌트 기술은 미시적으로 컴포넌트를 조립할 수 있는 구조, 관련 컴포넌트 및 이러한 구조를 분산 환경 하에서 유연하게 구현할 수 있는 미들웨어 프레임워크 등을 포함한다. 거시적으로는 컴포넌트 기반 시스템의 개발, 관리, 구현, 유지보수, 배포 등 소프트웨어 라이프사이클 전반에 걸친 기술을 포함한다. 현재 컴포넌트 기술은 구현 뿐 아니라 관련 기술전반을 포함하는 포괄적인 표준구조를 정의하는데 그 초점을 맞추고 있다. 구현을 중심으로 컴포넌트 기술의 표준 구조를 정의하는 것은 시장적 측면에서 불가능할 뿐 아니라, 호환성 측면에서도 문제가 있기 때문이다. 컴포넌트 기술의 핵심인 기술 요소들 간의 포괄적 호환성을 유지하기 위해서는 이에 관련된 모든 요소들을 포함하는 표준 구조의 정의가 필수적이라 할 수 있다.

3. OMG의 MDA 표준화 동향

3.1 MDA

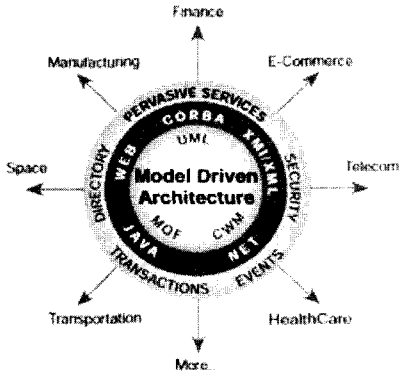
현재 컴포넌트 기술의 표준 구조는 OMG의 MDA(Model Driven Architecture)에 의해 제시되고 있다. MDA는 모든 컴포넌트 기술 요소의 표준 메타 모델(Meta Model)을 정의하고 이를 기반으로 각 구성요소를 정의함으로써 모든 컴포넌트 기술 요소들의 호환성 및 시스템 간 동작성을 보장하고 있다[1].

MDA의 핵심은 메타 모델을 기반으로 구현 환경에 독립적 모델(Platform Independent Model : PIM)을 자동으로 각 구현 환경에 적합한 구현 종속 모델(Platform Specific Model : PSM)로 변환 할 수 있는 구조를 정의하는 것이다. 즉 표준 메타 모델을 기반으로 구현 환경에 독립적인 시스템을 개발하고 이를 자동으로 구현 환경에 배치함으로써, 구현 단계에서의 생산성 향상 뿐 아니라, 표준 메타 모델에 기반을 둔 시스템들의 일반적인 호환성을 기대할 수 있게 되는 것이다[2].

3.2 MDA 구조

3.2.1 핵심 기술

MDA은 다음(그림 1)과 같은 네 가지 핵심기술을 근간으로 하여, 기본 서비스(Pervasive Services) 및 도메인 특성(Domain Facility)들로 구성된다.



(그림 1) Model Driven Architecture

1) MOF(Meta Object Facility)

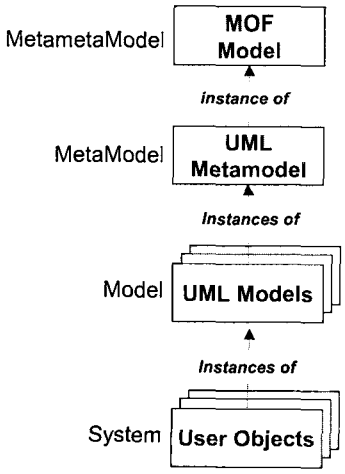
MOF는 객체 및 컴포넌트 기술의 핵심을 정형화한 모델로써, 객체지향 본질 그 자체이다[3]. MOF

는 객체지향 모델을 작성하기 위해 사용되는 메타 모델의 필수 요소와 문법, 구조를 정의하는 메타메타 모델이다. MDA에서 MOF는 CWM(Common Warehouse Metamodel)이나 UML(Unified Modeling Language) 메타모델에 대한 공통 모델로써 제공된다. OMG에서 제공하는 MOF 사양은 다음과 같은 내용을 포함한다.

- 일반적인 MOF 객체 및 관계에 대한 추상 모델
- MOF 기반의 메타모델과 언어 독립적인 인터페이스(CORBA IDL로 정의됨) 간의 매핑 규칙
- MOF 기반의 메타모델로부터 생성된 모델을 다루기 위한 일반적인 오퍼레이션들

2) UML(Unified Modeling Language)

UML은 객체 및 컴포넌트 시스템을 표현하기 위한 표준 언어로써 시스템의 아키텍쳐 및 구성 객체, 객체간의 상호작용 등을 모델링하기 위해 사용된다. UML은 시스템의 분석 및 설계 시에 구현환경에 무관하게 표준화된 방법으로 시스템을 모델링할 수 있게 한다. 각종 UML 다이어그램들은 UML 메타모델을 통해서 정교하게 정의가 되어있으며, 가시적인 UML 모델들은 다른 정형적인 언어로 자동 변경이 가능하다.



(그림 2) 4-계층 메타모델링 구조

(그림 2)는 MOF와 UML 모델들간의 관계를 나타낸다. 이러한 4계층 메타모델링 구조는 MDA기반 시스템 모델링의 기본 구조이다. MDA에서 UML은 구현 독립적인 모델이나 구현 종속적인 모델 작성 시에 사용되기도 하며, MOF나 CWM을 표현하기 위한 기호로서의 기반을 제공하기도 한다. MDA기반 시스템 개발의 첫 번째 단계가 UML을 이용하여 구현 독립적 모델을 작성하는 것이다.

UML은 각종 프로파일(Profile)에 의해 다양한 플랫폼에 대하여 구체적으로 특화 될 수 있다. 프로파일이란 UML의 기본 모델을 특화하거나 확장하기 위해 제공되는 메커니즘(태그 값, 스테레오 타입)을 말한다. OMG에서 정의한 CORBA나 EDOC (Enterprise Distributed Object Computing), EAI (Enterprise Application Integration) 등의 프로파일들은 구현 독립모델로부터 구현 종속 모델을 생성하거나, 구현 종속 모델로부터 코드를 생성하기 위한 규칙들을 제공하게 된다.

3) XMI(XML Metadata Interchange)

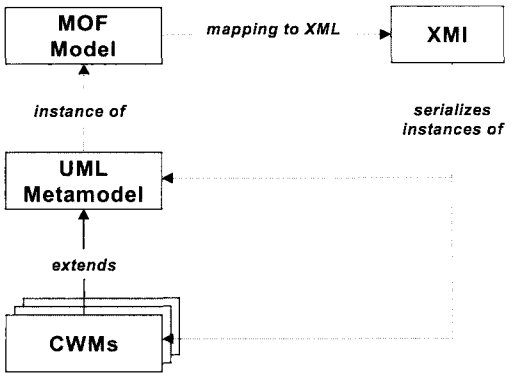
XMI는 XML(eXtensible Markup Language) 기반 데이터 관리를 위한 표준으로서 MOF기반 모델을 XML로 매핑하기 위한 표준 사양이다[4]. 구축된 모델이나 메타 모델들을 유용하게 사용하기 위해서는 이러한 모델들이 다양한 도구 간이나 혹은 도구와 리파지토리 사이에서 호환이 가능해야 한다. 이러한 것을 가능하게 하기 위해 OMG에서 표준화한 것이 XMI이다. UML과 MOF, XML에 기반하는 XMI는 DTD(Document Type Definition)와 스키마를 통해 XML로 메타 모델과 모델의 표현(Representation)을 표준화한다. XMI DTD와 스키마는 모델들에 대한 메타 데이터가 된다.

4) CWM(Common Warehouse Metamodel)

CWM은 데이터 웨어하우징이나 비즈니스 분석 영역에서 주로 나타나는 비즈니스 메타데이터와

기술적 메타데이터를 표현하는 표준 메타모델이다[5]. CWM은 이질적인 멀티 벤더 소프트웨어 시스템 간에 메타데이터 인스턴스를 교환하기 위한 기반을 제공한다. CWM은 자료 저장소(Data Repository) 통합을 위한 산업표준이며, 데이터베이스 모델(스키마), 스키마 변형 모델, OLAP(On-Line Analytical Processing), 데이터 마이닝 모델들의 표현방법을 표준화한다.

MOF와 CWM은 어플리케이션과 데이터 모델링, 결과를 저장하기 위한 리포지토리를 위한 기반을 형성하며, UML은 어플리케이션 모델을 구축한다(CWM은 그중에서도 데이터 모델을 구축하는 방법을 정의한다). XMI는 툴과 리포지토리 사이에서 모델의 호환을 담당한다(그림 3).



(그림 3) MDA 핵심기술간의 관계

3.2.2 기본 서비스(Pervasive Service)

분산 어플리케이션 구현을 위한 모든 플랫폼들은 공통적인 몇몇 필수 서비스를 제공하고 있다. 이러한 서비스들의 종류는 다양하지만 일반적으로 디렉토리 서비스(Directory Service), 이벤트 핸들링(Event Handling), 지속성(Persistence), 트랜잭션(Transaction), 보안(Security) 등을 포함한다. 이러한 서비스들이 특정 플랫폼에 맞춰 정의되거나 구축될 경우 해당 플랫폼에 한정되는 특성을 갖게 된다. 이러한 것을 피하기 위해 OMG는 UML의 구현 독립 모델(PIM) 레벨에 해당하는 기본 서비스를 정의하였다. 구현 독립 모델 작성 시 기본 서비스의 내용을 모델에 반영하며, 구현 종속 모델 작성 시 MDA에 의해 지원되는 모든 미들웨어 플랫폼에 대한 내용이 반영된다.

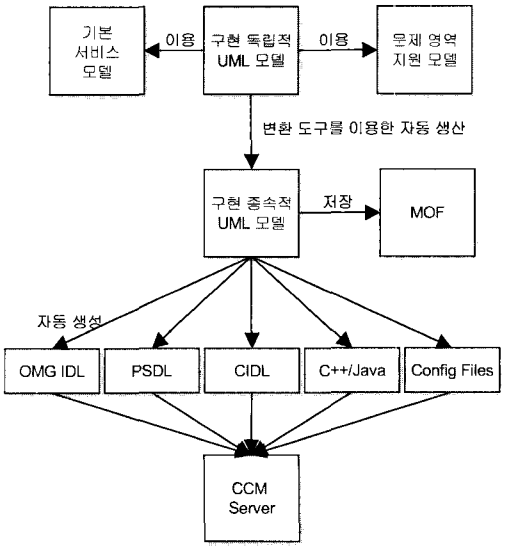
3.2.3 도메인 사양(Domain Specification)

OMG활동의 많은 부분이 DTC의 DTF(Domain Task Force)를 통해 특정 도메인 시장을 위한 서비스나 특성(Facility)을 표준화하는데 초점을 두고 있다. MDA에서 OMG의 도메인 특성에 대한 명세는 UML을 사용하여 표현된 구현 독립 모델(PIM)의 형태이다. DTC의 DTF는 해당 어플리케이션 분야의 표준 프레임워크를 정의한다.

3.3 MDA 기반 개발 공정

MDA기반 개발 공정은 자동화 도구를 기반으로 다음(그림 4)과 같은 단계로 이루어진다[6].

- 1단계 : 구현 독립적 모델 작성
- 2단계 : 구현 종속적 모델 작성
- 3단계 : 어플리케이션 생성



(그림 4) MDA 기반 CCM 서버 생성 단계

3.3.1 구현 독립적 모델(Platform Independent Model : PIM)

모든 MDA 기반 프로젝트는 (그림 4)의 상단에서 보는바와 같이 UML을 이용하여 구현 독립적 모델(PIM)을 작성하는 것으로부터 시작된다. 구현 기술과는 무관한 비즈니스 기능과 행위를 반영하는 MDA의 구현 독립적 모델은 시스템 프로그래머보다는 비즈니스 영역 전문가에 의해 구축되는 것이 바람직하다.

구현 독립적 모델 작성 시에 기본 서비스(Pervasive Service) 및 도메인 특성(Domain Facility)에 대한 내용을 포함하게 되는데, 일반적으로 MDA 어플리케이션 모델링 자동화 도구들이 이러한 기본 모델을 제공하게 된다.

3.3.2 구현 종속적 모델(Platform Specific Model : PSM)

구현 독립적 모델 작성이 완료되면 MOF로 저장되고 이 내용은 구현 종속적 모델을 생성하기 위한 매핑 과정의 입력물로 사용된다. 매핑 자동화 도구(Mapping Tool)는 구현 독립적인 모델을 구현 종속적(Platform Specific) UML 모델로 변환한다. UML에서 제공하는 제한 및 확장 메커니즘은 MDA에 의해 요구되는 구현 독립적 모델 및 구현 종속적 모델의 세부적인 표현을 가능하게 한다. UML 프로파일(Profile)이라 일컬어지는 표준화된 확장 메커니즘의 집합(스테레오 타입과 태그 값)은 특정 환경이나 특정 플랫폼 등의 한정된 사용을 위해 UML 모델을 특화할 수 있도록 한다. CORBA기반의 UML 프로파일은 2000년에 OMG에 의해 채택되었다. 현재 기타 플랫폼들에 대한 프로파일의 채택작업이 진행 중이다.

3.3.3 어플리케이션 생성

자동화 도구(Code Generation Tool)를 이용하여 구현 환경에 적합한 프로그램을 만들어 낸다. 구체적인 기능 등은 직접 프로그래밍 한다. 뿐만 아니라 MDA기반 자동화 도구들은 특정 플랫폼에서 요구하는 모든 파일을 생성하게 된다.

3.4 MDA 장점

MDA기반 시스템 개발 공정에서 알 수 있듯이, MDA는 메타 모델을 이용하여 대부분의 구현 공정을 자동화할 수 있는 구조를 제공한다. 이러한 MDA기반 개발 방법은 시스템의 구현 결과 뿐 아니라, 분석 설계 관리 등 프로젝트 진행 전체 결과를 재사용 가능하게 한다. 또한 시스템이 구현 환경과 독립적으로 정의되므로, 이식성(Portability)을 증가시킨다. 시스템 구조가 구현환경으로부터 독립됨으로서 개발된 시스템의 라이프사이클이 구현 종속적인 시스템에 비해 증가한다.

4. 결 론

컴포넌트 기술의 도입은 소프트웨어 생산성을 획기적으로 향상시킬 수 있는 유일한 방안으로 대두되고 있다. 하지만 컴포넌트 기술의 도입이 컴포넌트를 몇 개 만든다고 해결될 일이 아니다. 앞서 언급한 바와 같이 구조적 통합적 기술이므로, MDA 등과 같은 표준구조를 기반으로 점진적으로 장기적 안목을 가지고 추진되어야 한다. 즉 컴포넌트 기술의 도입 또한 CMM(Capability Maturity Model) 혹은 SPICE(Software Process Improvement and Capability dEtermination) 등의 소프트웨어 프로세스 평가 모델이 제시하는 바와 같이 단계를 거쳐 도입되어야 한다. 컴포넌트 기술의 도입은 조직의 기술적, 관리적, 재정적, 문화적 문제를 수반하기 때문이다. 참고적으로 Butler Group이 제안한 컴포넌트 기술 도입의 단계는 다음과 같다.

- 1단계 : 사용자 인터페이스 컴포넌트 정도의 이용 단계
- 2단계 : 구현 환경에서의 컴포넌트 이용 단계

- 3단계 : 분석 및 업무에서의 컴포넌트 이용 단계
- 4단계 : 컴포넌트 재사용의 단계
- 5단계 : 기업 업무 전반에 걸친 컴포넌트 이용 단계
- 6단계 : 성숙 단계

현재 국내에서는 KCSC(Korea Consortium for Software Component Promotion)를 중심으로 컴포넌트 기술을 적극적으로 도입하려는 시도가 이루어져, 관련 기업들이 2단계로 진입하려하고 있고, 미국의 경우는 OMG를 중심으로 많은 기업들이 3-4단계로의 진입을 시도하고 있다. 이러한 상황을 고려해 볼 때 국내 컴포넌트 기술의 조속한 정착을 위해서는 MDA등과 같은 컴포넌트 기술의 표준구조를 기반으로 미국 등과 같은 선진국들이 겪은 시행착오를 최소화하여야 할 것이다. 또한 컴포넌트 자체의 개발보다는 컴포넌트 구조를 중심으로 점진적, 장기적 기술 도입이 진행되어야 할 것이다.

참고문헌

[1] OMG Model-Driven Architecture Home Page: http://www.omg.org/mda/index.htm

[2] OMG Architecture Board MDA Drafting Team, "Model-Driven Architecture: A Technical Perspec - tive", ftp://ftp.omg.org/pub/docs/ab/01-02- 01.pdf

[3] OMG Meta Object Facility Specification, Version 1.3, September, 1999. http://www.dstc.edu.au/Research/Projects/MOF/rtf/. http://www.omg.org/.

[4] Object Management Group, XML Metadata Interchange Specification, Version 1.1, http://www.omg.org/.

[5] Tolbert, D., "CWM: A Model-Based Architecture for Data Warehouse Interchange", Workshop on Evaluating Software Architectural Solutions 2000, University of California at Irvine, May, 2000. http://www.cwmforum.org/uciwesas2000.htm

[6] Jon Siegel, "Developing in OMG's Model-Driven Architecture", ftp://ftp.omg.org/pub/docs/omg/01-12-01.pdf

저자약력

최 성 운

1985년 한국외국어대학교(상학학사)
1988년 미국 오레곤주립대학교(공학석사)
1992년 미국 오레곤주립대학교(공학박사)
1993년~현재 명지대학교 컴퓨터공학부 부교수, OMG KSIG 회장, 한국정보컨설팅(KIC) 기술자문
관심분야 : 컴포넌트 프레임워크, 객체지향 소프트웨어 공학
e-mail : choisw@mju.ac.kr

장 진 호

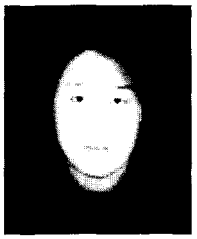
1976년 서울대학교 화학공학과(공학사)
1978년 KAIST 화학공학과(공학석사)
1978년~1982년 국방과학연구소
1983년~1985년 한국화약 대리
1985년~1998년 시스템공학연구소 책임연구원
1998년~현재 한국전자통신연구원 소프트웨어공학연구부 책임연구원
2001년~현재 TC09 간사 및 SG09.02 의장
관심분야 : 소프트웨어 공학 기술 표준, 컴포넌트 기반 웹 서비스, MDA기반 시스템 개발
E-mail : jhjang@etri.re.kr

전 인 걸

1996년 성균관대학교 정보공학과 (공학사)
1998년 성균관대학교 정보공학과 (공학석사)
1998년 시스템공학연구소 연구원
1998년~현재 한국전자통신연구원 소프트웨어공학연구부 연구원
2001년~현재 TC09의 SG09.01의 간사
관심분야 : 소프트웨어 품질평가, 소프트웨어 컴포넌트, 지능형 에이전트 시스템
e-mail : igchun@etri.re.kr

김 길 조

1987년 서울대학교 산업공학과 졸업 (공학사)
1989년 한국과학기술원 경영과학과 졸업 (공학석사)
1995년~1998년 시스템공학연구소 선임연구원
1998년~현재 한국전자통신연구원 소프트웨어공학연구부 선임연구원
2000년~현재 TC09 및 SG09.02 위원
관심분야 : 소프트웨어 품질평가, 소프트웨어 프로세스
e-mail : kgj@etri.re.kr



홍 선 주

1997년 명지대학교 컴퓨터공학과 졸업 (공학사)
1999년 명지대학교 컴퓨터공학과 대학원 졸업 (공학석사)
1999년~현재 명지대학교 컴퓨터공학과 대학원 재학
관심분야 : 객체지향 소프트웨어공학, 컴포넌트 기술
e-mail : hongsj@mju.ac.kr