제 19 회 한국정보처리학회 춘계학술발표대회 논문집 제 10 권 제 1 호 (2003. 5)

MDA 환경을 위한 다이어그램 편집기 생성 방법

정양재*
*한국전자통신연구원
e-mail : cornor@etri.re.kr

A Method for generating diagram Editors for MOF Environment

Yang-Jae Jeong*
Electronics and Tlelecommunications Research Institute

요 약

MDA(The Model Driven Architecture)는 모델을 기반으로 시스템을 개발하는 방법이다. 현재 MDA 에서 제안하는 대표적인 모델링은 PIM, PSM 으로 두 종류의 모델링 영역이 필요하지만, 적용 플랫폼에 따라 다양한 PSM 이 필요하므로 모델링 영역이 확장된다. 또 MDA 가 발전하면서 PIM 과 PSM 이 상대적인 개념임을 인식하면서 모델링 영역은 더욱 더 늘어날 예정이다. 다양한 모델은 다이어그램으로 표현되고 이를 지원하기 위해 메타 편집 기능이 필요하다. 본 논문에서는 MDA 환경에 필요한 다양한 모델을 표현하기 위해 다이어그램을 생성하는 방법을 기술한다.

1. 서론

점점 복잡해지고 대형화되는 소프트웨어를 보다 효율적이며 효과적으로 만들기 위해 많은 노력들이 있었다. 이런 노력으로 모델링 관점으로 다양한 도메인 특성에 적합한 모델링 방법이 만들어졌으며, 구현 관점으로 네트웍, 보안, 분산 이벤트 핸들링, 트랜잭션, 지속성 등의 서비스를 간편하게 할 수 있는 플랫폼이 개발됐다. OMG 는 모델과 구현, 플랫폼 사이를 통합하기 위해 MDA(Model Driven Architecutre)[1]를 제안했다. MDA 는 OMG 에서 발표한 표준인 MOF, CWM, UML, XMI 를 기반으로 통합한 시스템 개발 방법론이라고 할 수 있다. 개발 과정에서 PIM(Platform Independent Model), PSM(Platform Specific Model)등의 모델링이 필요하다. MOF 메타 모델을 바탕으로 도메인에 적합한 여러 PIM 이 만들어지고, 플랫폼의 특성을 표현하기 적합한 여러 PSM 이 만들어진다. MDA 를 제안한 초기에는 PIM, PSM 으로 모델링 단계를 나눴지만 MDA 의 개념이 발전하면서 PIM 과 PSM 이 상대적인 개념임을 인식하고 있다. 즉 모델링 영역은 시스템의 분할 정도, 추상화 수준, 도메인 영역에 따라 확대되고 정도에 따라 상대적으로 PIM 과 PSM 이 분류된다[2].

본 논문은 모델링 영역의 모델을 편집하기 위한 메타 편집기의 생성 방법을 기술한다. 현재 많은 메타 편집기가 나와 있지만, MOF 기반의 메타 편집기는 나와 있지 않다. MOF 기반의 편집기의 모델은 MOF 표준을 따르기 때문에 편집기의 모델을 쉽게 통합 할 수 있을 뿐 아니라 MOF 와 XML 을 따르는 다른 도구와도 쉽게 톱할 수 있는 장점을 가진다.

2. 연구배경

2.1 MDA(Model Driven Architecture)

MDA 는 OMG 가 지속적으로 작업해 온 모델링 관련 표준화 작업의 성과를 바탕으로 모델 중심의 시스템 구현을 위한 기술 구조를 정의한 것이다. MDA 의 핵심 사항은 시스템의 설계 및 명세를 구현 기술에 독립적인 모델로 기술하고 이를 자동으로 변환함으로 실제 구현 환경에 배치하는 것이다. 다시 말하면

MDA 는 시스템 통합과 유지를 위해서 PIM 으로부터 기술 종속적인 모델을 얻어내는 것이 가능하게 하는 기술 조건에 대한 정의 및 그에 대한 구조에 관한 것이라고 할 수 있다.

MDA 는 다음과 같은 네 가지 모델 관련 OMG 표준에 기반을 두고 있다.

- UML (Unified Modeling Language)[9]: 객체 및 컴포넌트 시스템을 표현하기 위한 표준언어이다.
- MOF (Meta Object Facility)[10]: 모델 정보에 대한 표준적인 저장소를 제공하고 표준화 된 방식으로 모델 정보를 접근하는 구조를 정의한다.
- CWM (Common Warehouse Metamodel)[11]: 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)[12]: UML로 기술된 모델 정보의 XML 표현에 대한 표준이다.

이러한 표준을 통해서 시스템 설계를 표준적인 방법으로 수행할 수 있으며 모델의 생성, 배포, 변환, 저장 등의 모델 관리도 정형화 된 방식으로 수행할 수 있다. 또한 미들웨어 공통의 디렉토리(Directory), 보안(Security), 분산 이벤트 핸들링(Distributed Event Handling), 트랜잭션(Transaction), 지속성(Persistence) 서비스 등의 기본 서비스(Pervasive Service)와 공통 도메인 특성(Domain Facility)을 PIM 수준에서 정의함으로 플랫폼 공통 서비스나 도메인 공통 모델이 특정 구현 기술에 종속되어 정의 되는 것을 피하고 구현 환경에 독립적으로 최대한의 정보를 PIM 수준에서 기술할 수 있게 하였다. 이러한 MDA 의 구성 기술 요소간의 관계는 (그림 1)이 잘 표현하고 있다.

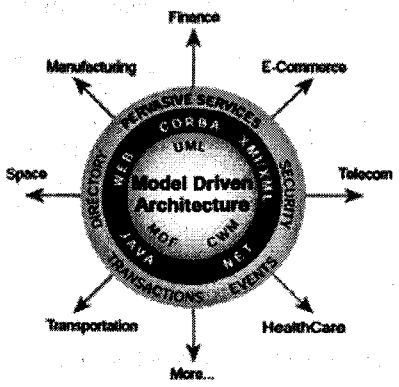

그림 1. Model Driven Architecture

2.2 UML Profile

UML profile 이란 UML 의 기본 building block 을 UML 에 자체의 확장 메커니즘인 Stereotype, Tagged Value, UML 의 Constraints 기술 방법을 이용하여 UML 을 특별한 목적에 맞게 변경한 것이다.

MDA 에서 UML profile 이 쓰이는 경우는 다음과 같다.

- PIM 수준에서 특정 도메인에 해당하는 노테이션들을 표현하기 위함 (예, UML profile for EDOC)
- PSM 수준에서 해당 기술 플랫폼을 UML 로 올바르게 표현하기 위함(예, UML profile for CORBA, UML profile for EJB, UML)

2.3 메타 편집기

도메인 환경과 플랫폼에 따라 생성된 프로파일을 활용하기 위해서는 각 프로파일에 대한 또는 뷰포인트에 대한 다이어그램 편집기가 필요하다. 메타 편집기는 다양한 다이어그램 편집기 생성을 지원하며 다음과 같은 장점을 갖는다. [3]

- 메타 편집기는 도메인 특성에 따라 모델링 하므로 전체 개발 시간을 단축할 수 있다. 새로운 개발 언어를 개발해도 구현 시간은 그리 빨라지지 않았다. 자바가 BASIC 보다 20%의 속도 증가를 보이는 반면, 도메인 특성에 맞는 언어를 사용할 경우 5-10 배의 속도 증가를 가져온다.
- 메타 편집기는 요구 사항 변경에 빠르게 대응한다. 구현 도중 요구사항이 변경되면 이를 구현에 반영하기 어렵다. 도메인 특성에 맞는 언어는 도메인 레벨이므로 쉽게 요구사항을 반영할 수 있다. 도메인 특성에 맞는 언어에서 최적화된 코드를 자동 생성한다.
- 도메인 전문가의 능력을 활용할 수 있다. 도메인에 대한 지식이 필요한 프로젝트에서 모든 사람이 도메인에 대한 지식을 습득하려면 많은 시간이 필요하다. 팀 내에 도메인 전문가가 도메인 특성에 맞는 언어를 개발하면 다른 팀원이 이를 활용하여 쉽게 모델링 할 수 있다. 모델링 언어로부터 자동 코드 생성을 지원함으로 구현을 용이하게 한다.

대표적인 메타 편집기는 MetaEdit+ MetaCase[3]와 DOME(the Domain Modeling Environment)[4]이 있다.

MetaCase 는 GOPRR (Grahp, Object, Property, Relationship, and Role)[3]이라는 메타모델링 언어를 사용한다. GOPRR 을 이용한 모델링은 이름에 담겨진 의미에 따라 Graph, Object, Property, Relation, Role 으로 이루어졌다. 개발자가 GOPRR 을 이용하여 메소드를 설계하고, 메타모델의 개념을 규칙과 노테이션을 설계한 후 생성기를 위해 에디터를 생성하면 다이어그램 편집, 모델 브라우징, 멀티 유저환경까지 지원하는 케이스 도구(CASE tool)가 생성된다. 현재는 MOF 를 GOPRR 의 서브타입으로 하여 MOF 메타모델링을 지원한다.

DOME 은 객체 모델링에 기반을 둔 메타 편집기이다. DOME 은 UML, Coad-Yourdon OOA, Colbert OOSD, IDEF0, Petri-Nets 등을 모델링을 지원하며, 가장 큰 특징은 새로운 노테이션을 추가하여 새로운 CASE 도구를 생성하는 것이다. 생성된 도구는 그래픽 모델링과 코드 생성과 분석, 도큐맨테이션을 지원한다. 현재 DOME 은 GNU 에 의해 소스코드가 공개됐으면 Smalltalk 으로 구현되었다. [5]

위 두개의 메타 편집기는 독자적인 메타모델링 언어를 사용하므로 모델 관점에서는 MOF 의 따른다고 할 수 없다. Meatcase 가 MOF 를 지원하지만 GOPRR 을 기본으로 하여 MOF 를 타입 변환했기 때문에 모델까지 MOF 를 따르지는 않는다. 본 논문에서는 뷰와 모델 모두 MOF 를 따르는 방법을 사용한다.

3. 메타모델에 대한 다이어그램 생성

UML 1.4 스펙에는 UML 모델에 대한 메타모델과 다이어그램에 대한 표현이 기술됐다. 다이어그램은 Semantics 와 Notation 을 중심으로 메타모델과의 관계를 비정형적으로 기술했다. UML 에서 제안하는 다이어그램 외에 메타모델에 대한 다양한 뷰포인트를 생성하는 것이 가능하다. 그러나 메타모델 관점에서 뷰포인트를 지원하는 방법이 존재하지 않는다. UML 모델러들은 모델에 대해 다양한 뷰포인트를 제공하려고 시도했지만 관련된 메타모델을 구현할 수 없었고, 일관된 방법으로 뷰포인트를 필터링할 수도 없었다[6.]

MDA 에서는 MOF 메타모델을 이용하여 다양한 도메인에 대한 메타모델을 정의하여 도메인 특성에 맞는 모델링이 가능하다. 도메인에 특성에 맞는 모델링은 구현 속도와 개발 시스템의 품질을 높여준다[3] *MOF 를 위한 메타 편집기는 MDA 환경의 다이어그램을 생성하여 도메인 특성에 맞는 모델링을 하는 편집기를 생성한다.* 메타 편집기는 다음과 같은 기능을 수행한다. 첫째, MOF 메타모델을 이용하여 메타모델을 정의한다. 둘째, MOF 리파지토리를 이용하여 모델 관리기를 생성한다. 셋째, 뷰포인트와 관련된 모델을 정의하여 다이어그램 모델을 정의한다. 넷째, 다이어그램에 대한 에디터를 생성한다.

3.1 메타모델 정의

도메인에 대한 메타모델 정의 방법은 크게 두가지로 분류된다. 첫번째 방법은 UML 의 메타모델에 UML 의 확장 방법을 이용하여 새로운 프로파일을 생성하는 방법이며, 두번째 방법은 MOF 모델을 이용하여 새롭게 메타모델을 정의하는 방법이다. UML 프로파일은 uml 의 확장 방법으로 확장이 제한되기 때문에 메타 모델에 대한 제한이 존재한다. MOF 메타모델 확장 방법은 MOF 의 풍부한 의미 표현을 할 수 있다. [6]

MOF 는 메타메타모델로서 다음과 같은 노테이션을 이용하여 메타모델을 정의한다.

- *Class – heading, super classes, contained element,* attributes, references, operations, constrains.
- Association
- Datatypes
- Exception
- Constants

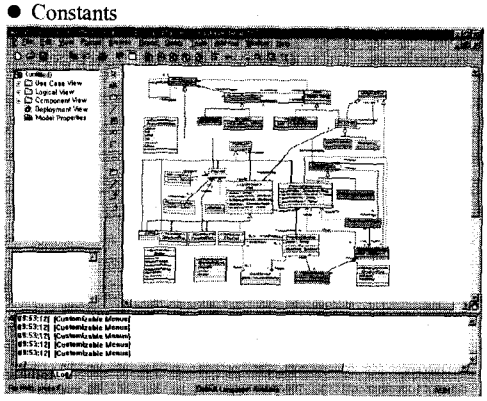

그림 2. profile4EDOC 의 CCA 프로파일

그림 2 는 profile4EDOC 의 CCA 프로파일에 대한 메타모델이다.

3.2 뷰포인트 정의

메타모델에 대한 모델은 뷰포인트에 따라 여러 다이어그램으로 표현될 수 있다. MDA 개발 환경에서는 뷰포인트에 따라 새로운 그래픽 노테이션을 정의할 수 있어야 된다.[6]. UML 의 경우 Behavioral Element, Foundation, ModelManagerment 의 세개의 패키지로 메타모델이 구성되어 있고 다이어그램은 여러 패키지 사이의 메타모델로 구성된다[7]. 따라서 다이어그램을 정의하기 위해 우선 다이어그램과 관련된 메타모델을 명시한다. 예를 들어 클래스 다이어그램은 클래스, 인터페이스, 패키지, 노트, 관계에 관련된 메타 모델을 갖게 된다. 다음에는 각 메타 모델이 뷰에서 표현하는 정보를 정의한다. 뷰에서 표현되는 모델은 메타 모델에 직접 정의한 내용과 상속, 포함 관계에 따라 다양한 정보를 포함한다. 노테이션의 모양은 편집기와 관련되기 때문에 다이어그램 모델에서는 기술할 수 없다. 메타 모델 classifier 의 서브 클래스로 정의된 클래스의 경우 내부에 정의된 애트리뷰트와 오퍼레이션 외에도 상속에 따라 애트리뷰트와 오퍼레이션을 포함한다. 그 중에 일반적으로 클래스 다이어그램에 표현하는 클래스 정보는 아래와 같이 세 부분을 나뉜다.

- name compartment : heading, superclasses, contained element
- attribute compartment : attributes, references
- operation compartment : operations, constraints

뷰포인트 모델에는 모델을 접근하기 위한 JMI, XMI 표준을 포함한다. 최종적으로 뷰포인트 모델은 뷰포인트에 표현할 모델들과 모델들의 접근 API 를 포함한

다. 뷰포인트 모델은 다이어그램 편집기 생성을 위해 편집기 생성 정보로 활용된다.

3.3 편집기 생성과 모델 관리

편집기 생성 과정에서 뷰포인트 정의 과정에서 생성된 뷰포인트 모델에 대한 편집기를 생성한다. 뷰포인트 모델을 독립적인 단위로 취급함으로 하나의 뷰포인트에 대해 사용자에 따른 여러 편집기를 생성할 수 있다.

생성된 편집기는 MVC(Mdoel-View-Control)패턴을 따라 구현된다. 모델은 MOF 리파지토리를 통해 자동으로 생성된다. MOF 리파지토리는 MOF 메타 모델에 대한 모델을 관리할 수 있는 모델 관리기를 자동으로 만든다. 또한 JMI[8]를 사용하기 때문에 표준 인터페이스를 이용하여 모델을 활용할 수 있다. 뷰와 컨트롤은 메타 편집기에서 생성되는 편집기가 담당한다. 편집기는 모델을 노테이션으로 표기하고 모델을 생성, 변경, 삭제 하는 등의 기능을 수행한다. 편집기의 기본 기능은 다음과 같은 JMI 의 표준 인터페이스를 통해 수행한다..

- 모델 생성 : RefClass.refCreateInstance
- 모델 삭제 : RefObject.refDelete
- 관계 생성 : RefAssociation.refAddLink
- 관계 삭제 : RefRemoveLink
- 패키지 생성 : Package.create
- 패키지 삭제 : RefPackage.refDelete

그림 3 은 메타모델을 편집하고 이에 대한 모델 관리기와 편집기를 생성한 후 연결한 모양이다.

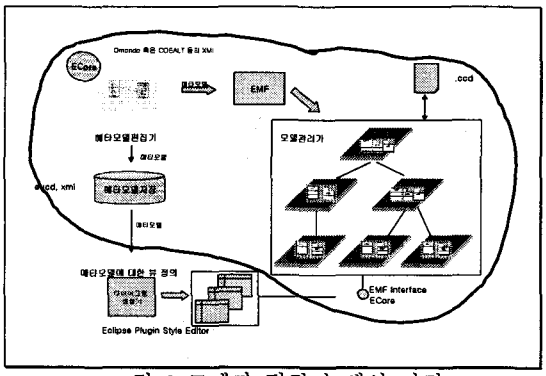

그림 3 모델과 편집기 생성 과정

4. 결론 및 향 후 일정

본 논문은 MDA 의 MOF 에 따른 모델들을 모델링하기 위한 편집기 생성 방법을 기술했다. 지금까지 개발된 메타 편집기들은 각 메타 편집기의 독특한 방법으로 편집기에 모델 정보를 모델링하고 편집기를 생성했다. MOF 를 기반으로 하는 메타 편집기는 MDA 환경에 필요한 수많은 모델을 위한 편집기를 쉽게 생성할 수 있으며, MOF 표준을 따르기 때문에 MOF-XMI, MOF-JMI, MOF-IDL 등의 표준 변환을 그대로 이용하여 모델 변환에 유리한 장점을 갖는다.

참고문헌

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

[2] Duane Hybertson. " Strengthening the Modeling Foundation of the MDA", Paper for Workshop in Software Model Engineering, 1 Octorber 2002.

[3] http://www.metacase.com

[4] Iyytinen, K. S. and Rossi, M. METAEDIT+ -- A Fully Configureable Multi-User and Multi-Tool CASE and CAME Environment. LGNS#1080, 1000. Springer-verlag.

[5] www.htc.honeywell.com/dome/description.gtm

[6] David S. Frankel. "Model Driven Architecutre applying MDA to Enterprise Computing:, David S.Frankel, Wiley Publinsing, 2003.

[7] Object Management Group. OMG Unified Modeling Language Specification v1.4. September 2001.

[8] Java Community Process. Java™ Metadata Interface(JMI) Specification version 1.0, 07-june-2002.