제22회 한국정보처리학회 추계학술발표대회 논문집 제11권 제2호 (2004. 11)

UML 2.0 Diagram Interchange Specification 지원 메타 에디터 프로토타입 개발

정양재, 신규상
한국전자통신연구원
e-mail : {cornor, gsshin}@etri.re.kr

Design Editor for UML 2.0 Diagram Interchange Specification

Yangejae Jeong0 Gusang Shin
Dept. of Office, ETRI

요 약

UML 은 1997 년 버전 1.0 이 나온 이후로 여러 변화를 거치면서 산업계의 표준으로 자리잡았다. UML 을 위한 다양한 툴이 개발되고 툴 사이의 모델 교환을 위해 UML XMI 이 개발 되었다. UML 2.0 에서는 모델 교환뿐 아니라 다이어그램 정보와 노테이션에 대한 정보도 교환하기 위해 다이어그램 교환을 위한 스펙을 발표했다. 다이어그램은 노테이션의 집합으로 이루어지고 노테이션은 다시 여러 하위 노테이션으로 구성된다. Diagram Interchange Specification 은 이런 관계를 XMI 로 표현한다. 다이어그램 정보 교환을 지원하는 편집기는 XMI 를 주고받을 수 있는 부분과 다이어그램 구조에 따라 다이어그램과 노테이션을 생성할 수 있는 기능이 필요하다. 본 논문에서는 다이어그램 메타 모델을 이용하여 모델 중심의 메타편집기 프로토타입을 보여준다.

1. 서론

OMG 에서 모델 중심의 시스템 개발을 위해 MDA[1]를 발표한 이후 2003 년에 UML 의 새로운 버전을 최종 승인하였다. UML2.0[2]에서는 리얼타임 시스템과 워크플로 시스템을 모델링 할 수 있는 기능이 추가되었다. 그렇지만 모델링 기능 추가보다 메타모델의 다시 정의되었다는 점에서 큰 변화라고 할 수 있다. 이와 더불어 UML2.0 에서 다이어그램 교환을 위한 Diagram Interchange Specification[3]이라는 스펙을 발표하였다. UML 은 모델 중심의 언어이지만 다이어그램으로 표현된다. 다이어그램에 대한 표준 스펙으로 인해 UML 이 보다 명확한 의미의 의사소통 수단으로 사용될 수 있다. 본 논문에서는 MOF 기반 메타 편집기[4]에서 사용되었던 다이어그램 메타 모델과 Diagram Interchange Specification 의 다이어그램 메타 모델을 비교한 후 상호 보완하여 UML2.0 을 지원하는 메타 편집기 프로토타입을 개발하고자 한다. UML 의 다이어그램 메타 모델은 다이어그램과 노테이션 사이의 포함 관계, 노테이션 사이의 포함관계를 잘 정의하였지만, 툴로 개발될 경우에는 많은 정보가 추가되어야 한다. [4]의 다이어그램 메타 모델은 이에 대한 보완으로 사용된다.

본 논문의 구성은 다음과 같다. 2 장에서는 본 논문의 관련 연구 내용을 기술하고, 3 장은 UML2.0 을 지원하는 메타 편집기에 대해 서술하며, 4 장에서는 메타 편집기의 프로토타입을 소개한다. 5 장은 결론 및 향후 연구 방향에 대해 기술한다.

2. 관련연구

2.1 UML 2.0

1997 년에 OMG 에서 표준 객체 모델링 언어로 채택된 UML 은 표현 영역을 넓히면서 산업계의 표준 모델링 언어로 자리잡아왔다. OMG 에서는 UML 의 표현력과 메타 모델의 구조까지 변경하여 2003 년 최종 승인하였는데, 이것이 UML 2.0 이다. 버전 번호가 바뀐 것으로 알 수 있듯이 UML 20 은 많은 부분에서 변

경되었다. 리얼타임 시스템과 워크플로 시스템을 모델링 할 수 있으며, 컴포넌트 기반 개발을 지원한다. UML 2.0 은 Infrastructure, Superstructure, OCL(Object Constraint Language), Diagram Interchange 와 같은 네 개의 명세서로 이루어졌다[5].

2.2 UML 2.0 Diagram Interchange Specification

Diagram Interchange Specification 이 UML 2.0 에 새롭게 추가되었다. UML XMI 가 UML 의 모델링 내용을 교환하기 위한 표준이라면 [3]은 다이어그램에 대한 정보를 교환하기 위한 표준이다. 다이어그램과 내부 노테이션과의 관계, 노테이션 내부의 관계를 XML 로 표현하여 전달한다. 다이어그램 관계를 정의하기 위해 별도의 메타모델을 정의하였으며 노테이션 모양은 SVG(Scalable Vector Graphics)[6]를 이용하여 기술한다.

2.3 MOF[7]

MOF 는 OMG 에서 정의한 메타모델 구조로서 4 단계의 모델 레벨로 구성된다. MDA 의 환경에서의 다양한 모델들에 대해 MOF 가 중요한 이유는 다양한 개발 단계에서 사용되는 모델을 통합할 수 있기 때문이다. MOF 를 중심으로 XML 을 위한 MOF XMI, 코바 IDL 을 위한 MOF IDL 매핑이 정의되어 있으며, MOF 리파지토리를 위한 JMI, IDL 등의 표준 인터페이스 등이 정의되어 있어 다른 표준들과 상호작용이 쉽게 이루어진다. 또한 CWM 의 메타모델에도 MOF 가 이용함으로 DB 와의 통합도 용이하다.

2.4 MOF 기반 메타편집기

MOF 기반 메타편집기는 MDA 환경에서 필요한 다양한 도메인 공간을 지원하기 위한 메타편집기이다. MOF 기반 메타편집기에서는 다이어그램 메타모델을 정의하여 다이어그램 정보를 모델로 관리하고 이후 정보를 활용할 수 있는 방법을 제시하였다. 다이어그램 메타모델에 따라 노테이션 사이의 포함관계를 정의하고 노테이션 사이의 배치 정보를 그래프로 정의하여 레이아웃 구현에 사용한다.

3. UML 2.0 기반 메타편집기

3.1 UML Diagram Interchange specification

UML2.0 이 발표되면서 UML1.X 와는 다른 스펙이 발표되었다. 그 중에 하나가 다이어그램을 교환하기 위한 Diagram Interchange Specification 이다. 지금까지의 UML 은 모델 중심이었고, UML2.0 역시 Infrastructure, Superstructure, OCL 에 대한 스펙으로 잘 정의된 메타모델을 제공하고, 정형기법적인 방법을 사용하여 모델이 엄격한 의미를 갖도록 하였다. 그렇지만 UML 은 이름에서 의미하듯이 모델링 언어이며, 이 언어는 다이어그램을 통해 표현된다. 언어가 정확한 발음으로 인해 명확한 의사소통이 이루어지듯이 UML 도 통일된 다이어그램 표현법을 통해 툴 사이의 명확한 모델 교환이 가능해진다. 물론 다이어그램과 모델의 매핑이 명확하게 정의되어야 하는 것이 선행 조건이다.

3.2 다이어그램 메타 모델

[4]에서는 다양한 다이어그램의 정보를 표현하기 위해 다이어그램 메타 모델을 만들었다. 표준 메타 모델인 MOF 를 이용하여 다이어그램 메타 모델을 정의하였기 때문에 다이어그램 모델을 다양하게 활용할 수 있다. 그림 1 은 [4]에서 제안한 다이어그램 메타모델이다.

그림 2 는 UML 2.0 Diagram Interchange Specification 에 정의한 다이어그램 메타모델이다. MOF 를 사용하여 메타모델을 작성하였으므로 MOF XMI 를 적용하여 다이어그램 정보를 XML 로 표현할 수 있다. 다이어그램 정보를 포함한 XML 파일은 툴 사이의 다이어그램 교환 정보로 활용된다. [4]의 다이어그램 메타 모델은 메타 편집기를 개발하기 위해 노테이션 사이의 관계와 노테이션의 표현에 중심을 두었고, UML 2.0 의 다이어그램 메타 모델은 다이어그램과 노테이션과의 관계에 중심을 두었으며, 모델과의 연결 정보를 명확히 했다.

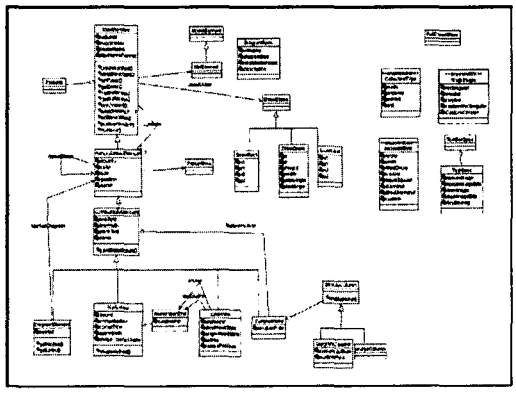

그림 1 메타편집기를 위한 다이어그램 메타모델

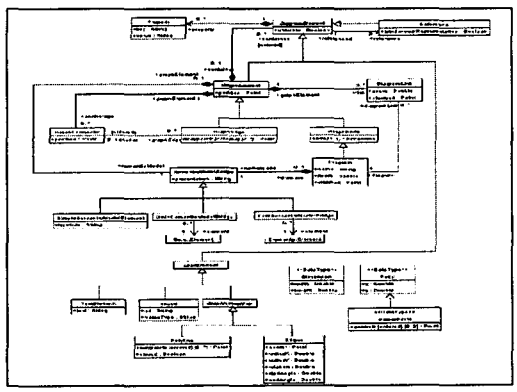

그림 2 UML2 의 다이어그램 메타모델

3.3 메타 다이어그램 편집기

메타 다이어그램 편집기는 그림 3 과 같은 절차로 생성된다. 메타 모델을 기반으로 다이어그램을 정의하고, 다이어그램 생성기를 통해 모델편집기를 자동 생성한다. 다이어그램 정의 과정에서 MOF 메타모델의 연결 정보를 기술하여 다이어그램 편집에 따라 명확한 모델이 생성된다. [4]의 프로토타입에서는 독자적인 다이어그램 메타 모델을 가정했었지만 본 논문에서는 UML2.0 의 표준 다이어그램 메타 모델에 대한 프로토타입을 개발하고자 한다. 그렇지만 UML2.0 의 표준 다이어그램 메타모델의 정보만으로는 메타편집기를 만들기 어려우므로 [4]의 다이어그램 메타모델을 혼합하여 사용한다.

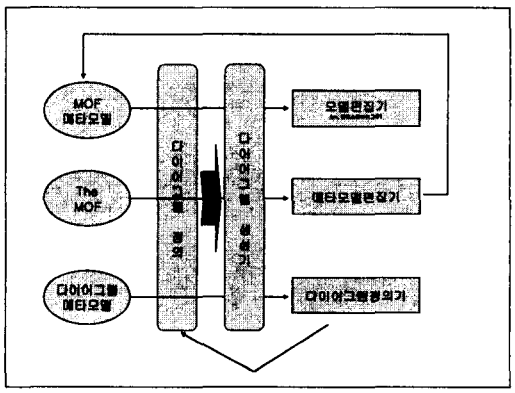

그림 3 메타편집기 생성 과정

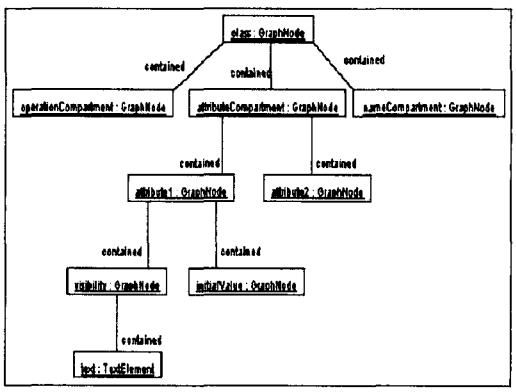

그림 4 UML 다이어그램 메타모델의 클래스 정의

그림 4 는 클래스 노테이션의 중첩관계를 정의한 다이어그램 모델이다. 클래스는 name, attribute, operation 을 위한 3 개의 부분으로 구성되어 있고 attribute 의 예를 통해 각 부분의 내부 구조가 더 있음을 표현한다. 그림 5 는 편집기를 생성하기 위해 클래스의 노테이션을 표현하기 위한 다이어그램 모델이다. 그림 4 와 동일하게 3 개의 부분이 있으며 각 부분에 대한 노테이션을 정의했다. 편집기 생성과정에서 각 부분이 조합되어 클래스 노테이션을 구성한다. 그림 6 은 노테이션과 관련 모델의 매핑을 표현한 그림이다. 노테이션의 편집에 따라 연결된 모델이 편집된다.

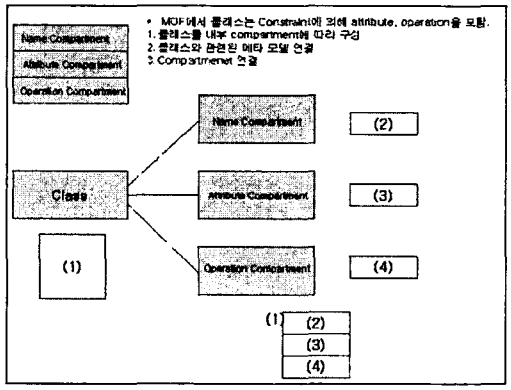

그림 5 메타 편집기를 위한 클래스 정의

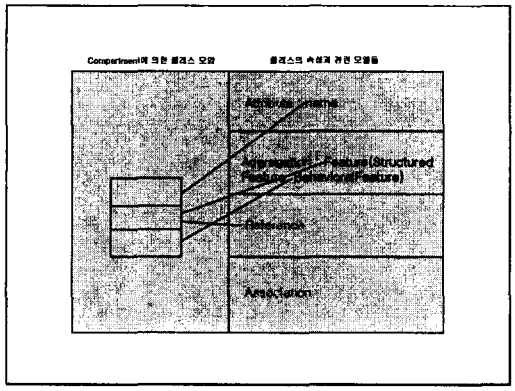

그림 6 노테이션과 모델과의 관계

4. 프로토타입

그림 4, 5, 6 과정을 통해 다이어그램 모델을 정의하여 노테이션의 중첩 관계를 표시하고 편집기 내에서 사용되는 노테이션과 관련 메타 모델을 정의한다. 다이어그램 편집기 생성기에서는 이 정보를 활용하여 다이어그램 편집기를 자동 생성한다. 관련 메타 모델이 MOF 메타 모델이기 때문에 JMI(Java Metadata Interface)와 같은 표준 API 를 사용하여 모델을 관리한다. 프로토타입은 ECLIPSE 기반으로 작성되었다. GEF 를 사용하여 ECLIPSE 플러그인으로 개발했다. 그림 7 은 클래스 다이어그램에 대한 프로토타입, 그림 8 은 유스케이스 다이어그램에 대한 프로토타입, 그림 9 는 EDOC[8] 프로파일 중의 하나인 CCA 에 대한 프로토타입으로 메타편집기에 의해 자동 생성된다. 메타 편

집기 시스템의 구조와 기능은 [4]에 기술되어 있다.

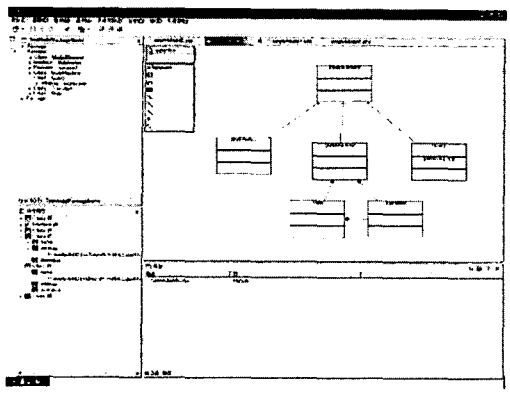

그림 7 클래스 다이어그램 프로토타입

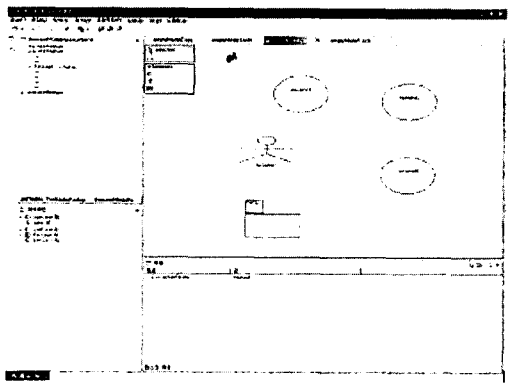

그림 8 유스케이스 다이어그램 프로토타입

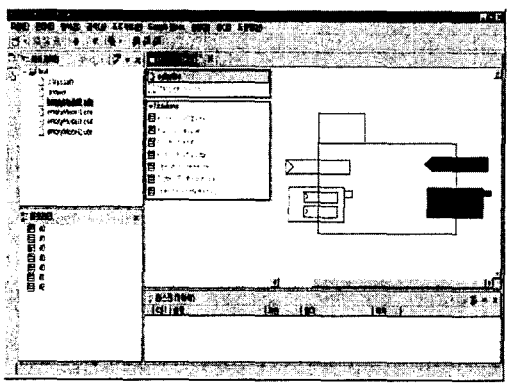

그림 9 CCA 다이어그램 프로토타입

5. 결론

본 논문에서는 메타 모델 기반 메타편집기의 프로토타입을 개발하였다. UML2 의 표준 스펙으로 발표된 Diagram Interchange Speciation 의 다이어그램 메타모델과 [4]의 다이어그램 메타모델을 이용하여 다이어그램을 정의하고 그에 대한 편집기를 생성하였다. 프로토타입에서는 다이어그램 메타 모델에 대한 클래스를 직접 작성하여 구현했지만 추후 MOF 메타모델을 이용하여 다이어그램 메타모델을 생성할 예정이다. 다이어그램 메타모델은 UML2 와 [4]의 다이어그램 메타모델을 혼합하여 메타편집기 생성에 필요한 모든 정보를 포함해야 한다. 본 논문에는 기술하지 않았지만 다양한 노테이션의 위치 관계를 그래프 형태로 정의하여 노테이션의 표준 레이아웃 알고리즘을 정의했다.

참고문헌

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

[2] Object Management Group.:OMG Unified Modeling Language Specification. Ver 1.4, p. xxi, September 2001.

[3] Object Management Group.:UML2.0 Diagram Interchange Specification. Ver 2.0(2003)

[4] 정양재, 신규상.:MOF 기반 MDA 개발 환경 구축. KCSC 학술대회(2004).

[5] 김현남. :생각하며 배우는 UML2.0. 영진닷컴, (2004) 27

[6] Object Management Group.:UML2.0 Diagram Interchange Specification. Ver 2.0(2003) 1.

[7] Object Management Group.:Meta Object Facility(MOF) Specification. Ver. 1.4, (2002)

[8] Object Management Group.:UML Prifile for Enterprise Disributed Object Computing Specification. (2002)