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

컴포넌트 기반 MDA 모델 및 편집 도구에 관한 연구

이지현*, 윤석진, 신규상
한국전자통신연구원 S/W 컨텐츠기술연구부

e-mail : {jihyun, sjyoon, gsshin}@etri.re.kr

A Study on Model and Modeling Tool based on MDA

Jihyun Lee*, Seok-Jin Yoon, Gyu-Sang Shin
S/W Contents Technology Department, ETRI

요 약

다양해진 플랫폼으로 어플리케이션의 상호 운용성 및 이식성이 떨어지는 문제를 해결하고자 MDA(Model Driven Architecture)에 기반하여 모델을 구성하고 모델을 그래픽적으로 표현하기 위한 편집 도구의 방향에 대해 제안한다.

1. 서론

비즈니스 어플리케이션의 규모가 보다 대형화되고 개발 프레임워크의 종류가 다양해짐에 따라 어플리케이션 개발자들은 어플리케이션을 개발, 통합 및 유지 보수하는 작업을 보다 신속하게 진행할 수 있는 기술에 관심을 갖게 되었다.

컴포넌트화된 단위 소프트웨어를 개발하고 컴포넌트들간의 통합을 통해 사용자가 원하는 기능을 수행하도록 하는 컴포넌트 개발 방법론은 컴포넌트의 재사용을 높이고 소프트웨어의 유지 보수를 용이하게 하는 방법으로 사용되고 있다. 하지만, 컴포넌트 기반 개발 방법은 J2EE, .NET, CORBA 와 같은 다양한 컴포넌트 플랫폼이 등장함에 따라 설계 시 특정 플랫폼에 종속적으로 모델링 하게 되어 어플리케이션간의 상호 운용성 및 이식성이 떨어지게 되는 문제를 가져왔다. 이러한 문제는 새로운 플랫폼이 등장할 경우 기존의 소프트웨어를 재사용하기 위해 변환에 많은 어려움을 초래할 것으로 예상된다. 이에 대한 대안으로 OMG(Object Management Group)는 2001 년 9 월에 MDA(Model Driven Architecture)를 국제 표준으로 채택하여 소프트웨어 개발 플랫폼에 대한 영향을 해결할 수 있는 개발 방법을 제시했다.

MDA 는 플랫폼 독립적인 설계 모델로부터 매핑을 통해 플랫폼 종속 모델을 생성하고, 플랫폼 종속 모델로부터 코드를 생성할 수 있도록 하여 다양한 플랫폼에서 어플리케이션이 상호 통합되고 이식될 때 발생되는 문제를 해결할 수 있는 방법을 제공한다. 현재 MDA 를 지원하기 위한 UML(Unified Modeling Language), MOF(Meta-Object Facility), XMI(XML Meta-Data Interchange), CWM(Common Warehouse Meta-model) [1][4]의 기술이 계속 표준에 맞춰 확장되고 있다.

MDA 를 지원하는 도구는 플랫폼 독립 설계 모델(Platform Independent Model; PIM)이 특정 도메인에 대한 요소들을 플랫폼에 독립된 정보로 표시할 수 있도록 표준화된 방법을 지원하고, PSM 모델로 변환될 수

* 본 연구는 2003 년 과학기술부 국가지정연구실 사업으로 수행되었음.

있는 기능을 기본적으로 제공해야 한다[1][2][3].

본 논문에서는 MDA 에 기반한 모델을 구성하기 위한 요구 사항과 모델을 그래픽적으로 표현하기 위한 효율적인 편집 도구 생성 방법에 대해 제안한다.

2. 모델 구성 요소와 요구 사항

현재 OMG 의 MDA 를 기반으로 opengroup 에서 COMBINE 프로젝트[5]가 진행 중에 있다. COMBINE 프로젝트는 개발 방법, 모델링 방법, 개발 지원 도구 등으로 나눠 진행됐고, 올 해 결과를 발표할 예정에 있다. COMBINE 에서는 MDA 기반 모델을 통해 엔터프라이즈 수준의 비즈니스 시스템을 구축할 수 있도록 UML profile for EDOC[6]에 기반하여 표현한다. EDOC 프로파일은 다양한 관점에서 비즈니스 시스템을 모델링 할 수 있는 명세로서 각 관점에서 구성하는 모델링 요소를 다음과 같이 정의한다.

* Enterprise Viewpoint : 비즈니스 시스템의 구조, 객체(object), 역할, 프로세스에 관해 모델링 한다.
* Information Viewpoint: 시스템에서 관리되어야 하는 정보(entity)와 관계(relationship)를 모델링 한다.
* Computational Viewpoint : 컴포넌트의 구조와 제공하는 비즈니스 트랜잭션을 컴포넌트의 인터페이스로 모델링 한다.
* Technology Viewpoint: 위의 모델에 특정 구현 기술을 적용하여 변환한 모델을 표현한다.

본 논문에서는 enterprise viewpoint, information viewpoint, computational viewpoint 를 PIM 으로 모델링하고, Technology viewpoint 를 PSM 으로 모델링 한다.

2.1. PIM 구성 요소 및 요구 사항

PIM 모델은 개발하고자 하는 시스템의 기능적, 비기능적 요구 사항을 분석하고 구현 기술과 독립적으로 모델을 표현할 수 있어야 한다. 따라서, 시스템의 기능과 구조 정보를 명시하기 위해 PIM 모델에 반영되어야 하는 세부 구성 정보를 본 논문에서는 시스템 아키텍처, 컴포넌트, 프로토콜(protocol)로 구분한다. 다음은 각 세부 모델 표현을 위한 요구 사항을 분석한 것이다.

* 시스템 아키텍처
  - 아키텍처는 특정 도메인에 포함되는 기능을 새롭게 모델링 해야 하는 기능과 재사용 가능한 기능으로 나누어야 한다.
  - 새롭게 모델링 하는 기능은 구현이 없는 컴포넌트로 명시되고, 외부와 데이터를 교환하기 위한 인터페이스를 포트(port)로 가져야 한다.
  - 재사용 가능한 기능은 기존 컴포넌트의 기능을 조회하고, 원하는 인터페이스를 선택하여 재사용할 수 있어야 한다.
  - 컴포넌트들 간의 기능 호출 및 리턴 값의 반환은 프로토콜로 명시되어야 한다.

* 컴포넌트
  - 컴포넌트는 기능에 대한 명세와 포트를 가져야 한다.
  - 포트는 컴포넌트가 갖고 있는 각 기능을 외부에서 접근할 수 있는 인터페이스로 기술해야 한다.
  - 포트는 구현 레벨에서 메소드 시그너처와 일대일로 대응되어야 한다.
  - 컴포넌트는 비즈니스 로직을 포함하는 프로세스 컴포넌트와 퍼시스턴스 정보 관리를 위한 엔터티 컴포넌트로 나뉘어야 한다.

* 프로토콜
  - 프로토콜은 컴포넌트 간의 동작 흐름을 명시해야 한다.
  - 프로토콜은 호출 컴포넌트에서 보내는 메시지, 기능을 갖고 있는 호출되는 컴포넌트의 특정 포트, 반환되는 결과 값으로 구성되어야 한다.

이와 같이 PIM 은 3 개의 시스템 아키텍처, 컴포넌트 구조, 행위에 대한 명세로 나눠 표현했다. 각 세부 모델은 시스템에 대한 요구사항을 개발자가 직관적으로 식별할 수 있도록 다이어그램으로 표현하며, PIM 편집 도구를 통해 아키텍처 다이어그램, 컴포넌트 다이어그램, 액티비티 다이어그램으로 모델링 될 수 있어야 한다.

2.2. PSM 구성 요소 및 요구 사항

MDA 의 정책에 따라 완성된 PIM 에 정해진 변환 규칙을 적용하여 PSM 을 생성한다. PSM 은 구현 코드

로 변환될 수 있는 구현 환경 정보를 포함하기 때문에 플랫폼 종류, 구현 언어, 데이터베이스 등 실제 구현 코드 및 동작에 관련된 정보가 포함될 수 있도록 반복적으로 모델링 되어야 한다. 다음은 본 논문에서 PSM 을 정제하기 위한 요구 사항을 분석한 것이다.

* 엔터티 컴포넌트의 정제
  - 엔터티 컴포넌트에 데이터베이스의 종류, 접속 정보, 질의어를 표현할 수 있어야 한다.

* 코드 생성 정보의 정제
  - 컴포넌트가 물리적으로 포함될 패키지를 표현할 수 있어야 한다.
  - 컴포넌트의 포트에 명시된 파라미터 타입을 구현 언어의 특정 타입으로 지정할 수 있어야 한다.
  - 컴포넌트의 네이밍(naming)을 위한 고유 식별명을 명시할 수 있어야 한다.
  - 클래스나 물리적 파일의 이름을 정하기 위한 규칙을 지정할 수 있어야 한다.

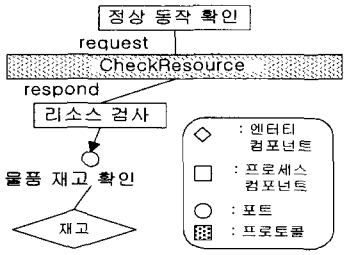
* 클라이언트 생성 정보의 정제
  - 컴포넌트의 인터페이스를 통해 수행될 특정 프로세스를 기술할 수 있어야 한다.

* 전개 정보의 정제
  - 어플리케이션 서버의 접속 정보를 명시할 수 있어야 한다.
  - 컴포넌트 상호 간의 참조 정보를 명시할 수 있어야 한다.

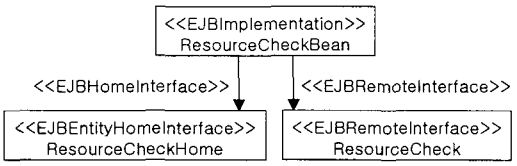
PSM 은 플랫폼에 기반한 정보를 표현할 수 있어야 하며, 플랫폼이 변경될 때 필요한 정보를 입력 받을 수 있도록 한다. PSM 은 코드와 1:1 로 대응되기 때문에 SRE(Simultaneous Roundtrip Engineering)이 가능할 수 있게 정제되어야 한다.

3. 모델 편집 방향

PIM 과 PSM 의 모델 정보는 UML 에 기반한 모델로 표현되고 시스템 아키텍트와 개발자로 하여금 특정 목적에 맞는 시각적인 모델 정보를 표현할 수 있도록 자체 확장 메커니즘인 stereotype, tagged value, constraints 를 사용한다. 본 논문에서 제시한 PIM 과 PSM 의 모델 구성 요소는 다음과 같이 표시될 수 있다. 그림 1 은 자판기 시스템의 리소스 관리 부분의 PIM 을 본 논문에서 분석한 요구 사항에 맞춰 보여주고, 그림 2 는 EJB 플랫폼이 적용된 '리소스 검사' 컴포넌트의 개략적인 PSM 구조를 나타낸다.



[그림 1] 자판기 시스템의 아키텍처 모델 부분 예



[그림 2] '리소스 검사'의 EJB 구현 예

도메인에 따라 PIM 과 PSM 에서 표현하고자 하는 노테이션과 의미 정보는 변경될 수 있다. 따라서 MDA 의 모델링을 지원하는 도구는 표현하고자 하는 모델 구성 요소와 의미 정보가 변경될 때 편집 도구가 쉽게 확장될 수 있도록 구성되어야 한다.

모델을 다이어그램으로 표현할 때 모델마다 독립적으로 편집기를 구성하는 방법은 표현하고자 하는 노테이션과 노테이션의 의미 및 행위 정보를 미리 만들어 놓고 모델을 편집할 수 있도록 하므로, 새로운 모델이 생길 때마다 새로운 편집 도구를 구현해야 하는 단점을 갖는다. 또한 편집기들을 통합하고 편집기 상호 간의 공유 모델 정보를 접근 및 관리하기 위한 인터페이스를 통일시키는 추가 작업이 필요하다.

독립 편집기 환경은 도메인에 따라 다양한 모델을 표현해야 함으로 모델을 표현할 기호와 의미 정보를 메타 모델로 입력 받고 편집기를 동적으로 생성할 수 있는 메타 편집 환경으로 확장될 필요가 있다. 메타 모델에 기반하여 모델링 구조와 의미가 동적으로 자동 변경되는 모델 편집기를 얻을 수 있다.

4. 메타 편집 지원 도구 메커니즘

메타 편집 도구는 모델링을 위한 메타 모델을 입력받아 메타 모델에 명시된 노테이션과 노테이션을 해석하기 위한 분석 엔진, 편집기를 동적으로 생성하기 위한 편집기 생성기로 구성된다.

* 메타 모델 작성기
  - 메타 모델은 모델링을 위한 노테이션과 의미 정보를 UML에 기반하여 모델링 한다.

* 메타 모델 분석 엔진
  - 메타 모델을 입력으로 받아 노테이션의 종류, 모양, 노테이션의 행위 정보를 분석하여 모델 저장소를 모델 저장소 생성기에 전달한다.

* 모델 저장소 생성기
  - 메타 모델에 대한 저장소를 구현한다.
  - 메타 모델에 접근할 수 있는 인터페이스를 편집기 생성기에게 전달한다.
  - 모델 저장소의 인스턴스에 모델 정보를 저장한다.

* 편집기 생성기
  - 모델 정보 저장소에서 생성된 인터페이스를 통해 메타 모델에 접근한다.
  - 다이어그램으로 모델을 그릴 수 있도록 그래픽 편집기를 구성한다.
  - 생성된 모델 정보는 모델 저장소를 이용하여 저장한다.

5. 결론 및 향후 연구

본 논문에서는 컴포넌트에 기반하여 MDA 모델을 표현할 때 모델로 구성해야 할 요소와 모델 편집을 위해 필요한 편집 도구의 방식을 결정하였다. 현재 메타 모델로 모델과 노테이션 정보를 정의하는 연구가 진행 중에 있고, UML 확장 프로파일을 이용한 모델 저장소를 생성하여 편집기를 동적으로 생성할 수 있는 편집기 생성기에 대한 연구가 진행 중에 있다.

참고문헌

[1] John D. Poole, "Model-Driven Architecture: Vision, Standards And Emerging Technologies," In Workshop on Metamodeling and Adaptive Object Models of the 15th ECOOP 2001, June 18-22, 2001.

[2] Jorg P. Wadsack and Jens H. Jahnke, "Towards Model-Driven Middleware Maintenance," OOPSLA 2002 Workshop, Nov. 4-8, 2002.

[3] Marie-Pierre Gervais, "Towards an MDA-Oriented MEthodology," pp. 265-270, COMPSAC 2002, Oxford, England, Aug. 26-29, 2002.

[4] MDA Specifications, http://www.omg.org/mda/specs.htm

[5] COMBINE Project, http://www.opengroup.org/combine/overview.htm

[6] UML profile for EDOC, http://www.omg.org/technology/documents/recent/omg_modeling.htm