EJB 어플리케이션 생성을 위한 MDA 변환 규칙 정의

이진열°, 라현정, 김수동
숭실대학교 대학원 컴퓨터학과
{jylee°, hjla}@otlab.ssu.ac.kr, sdkim@ssu.ac.kr

A MDA Transformation System for Building EJB Applications

Jin Yeal Lee°, Hyun Jung La, Soo Dong Kim
Dept. of Computer Science, Soongsil University

요 약

모델 기반 아키텍처 (Model Driven Architecture, MDA)는 플랫폼 독립적인 모델로부터 변환 규칙을 이용하여 특정 플랫폼 용 모델을 생성하는 소프트웨어 자동화 기술로 각광을 받고 있다. EJB(Enterprise JavaBeans)는 컴포넌트 기반의 분산 컴퓨팅을 위한 아키텍처로써 Java 기반 어플리케이션 개발에서 가장 널리 사용되는 개발 플랫폼이다. 기존의 PIM 에서 EJB 용 PSM 으로 변환 규칙에 대한 연구는 아직 미흡하고 체계적이지 못하다. 본 논문에서는 PIM 의 구조적인 구성요소와 EJB 용 PSM 의 구성요소를 비교 분석하여 변환 규칙을 정의한다. EJB 어플리케이션 개발을 위해 제안된 변환 규칙을 적용한다면 모델간의 대응관계를 효율적으로 표현 할 수 있기 때문에 이들간의 일관성과 추적성을 높일 수 있고 제품의 생산성, 유지보수성을 높일 수 있다.

1. 서론

OMG(Object Management Group)에서 국제 표준으로 채택한 MDA는 시스템 개발을 자동화 도구를 이용하여 상위 수준의 플랫폼 독립적인 모델(PIM)로부터 하위 수준의 구현 플랫폼 종속적인 상세 모델(PSM)로 변환 규칙을 적용하여 자동으로 변환하는 개발 아키텍처이다[1]. 모델간의 변환(Transformation)은 자동화와 함께 MDA에서 주요 기술 중에 하나이며 이러한 모델간의 변환은 변환 규칙을 기반으로 자동화된 툴에 의해 변환된다. 따라서 PIM에서의 수정만으로 전체 프로세스에 대한 유지보수가 가능하게 되고 개발자의 수고를 덜 수 있게 된다. 그러나 기존의 모델 변환규칙은 연구가 아직 미비하고 구조화 및 체계화 되어 있지 않다.

EJB(Enterprise JavaBeans)는 Java 기반 어플리케이션 개발에서 가장 널리 사용되는 개발 플랫폼이다. MDA에서 EJB 기반의 어플리케이션을 개발하기 위해서는 PIM에서 EJB용 PSM으로의 변환 규칙이 필요하다. EJB 어플리케이션 개발을 위한 변환규칙을 체계적으로 정의하고 PIM단계에서 이를 적용한다면 EJB용 PSM으로의 변환이 용이 할 것이다.

본 논문에서는 PIM과 EJB를 위한 PSM을 비교, 분석함으로써 모델간의 변환 규칙을 정의한다. 제안된 변환 규칙을 PIM에서 EJB를 위한 PSM변환에 적용하면 모델간의 일관성과 제품의 생산성, 일관성 및 유지보수성을 높일 수 있다.

2. 기반 연구

2.1. MDA

MDA는 소프트웨어의 개념적 설계 모델을 명세하고, 이를 상세 설계 모델과 코드로 변환하는 새로운 개발 기술이다. MDA에서는 두 개의 핵심적인 모델을 사용한다. PIM은 구현 기술의 독립적인 추상화 수준이 높은 모델이고 PSM은 PIM에서 변환 규칙을 이용하여 툴에 의해 자동적으로 변환된 구현 기술을 적용한 모델이다.

따라서 PIM에서의 수정만으로 유지보수가 가능하게 되고 기존의 개발자가 직접 코드를 수정하는 시간과 비용의 수고를 덜어주게 된다. 그리고 모델간 변환된 산출물과 설계가 일관성을 가지게 되므로 산출물간의 추적성을 높일 수 있다.

2.2. EJB 를 위한 UML 프로파일

UML 프로파일은 UML 기반의 설계를 위해 필요한 추가적인 명세의 표준장치들로 구성된다[3]. 이러한 UML 프로파일은 스테레오타입(Stereotype)과 태그 값(Tagged Value)이 적용된 엘리먼트(Element), 애트리뷰트(Attribute), 메소드(Method), 링크(Link) 로 구성된 표준 확장 메커니즘이다.

EJB 프로파일은 EJB 아키텍처를 기반으로하는 소프트웨어 시스템과 이를 기반으로하는 산출물을 UML의 확장 메커니즘을 이용하여 프로파일에 정의된 표준 표현 방식을 빌어 기술된다.

MDA 개발에서 각 모델간의 자동적인 변환을 위해서는 설계 모델이 UML프로파일에 의해서 명세화 되어야 하고 이를 이용하여 변환 규칙을 정의하고 PIM에서 PSM의 변환에 적용하게 된다.

3. 플랫폼 독립적인 모델의 구성요소

PIM은 최종 구현 플랫폼의 특정 정보도 가지고 있지 않은 모델을 말한다. 이러한 PIM은 추상화 수준이 높기 때문에 PIM에서 PSM으로 변환 규칙을 이용하여 어떠한 형태의 플랫폼으로도 변환이 가능하다.

PIM의 표현 방법은 다양하지만 본 논문에서는 PIM단계에서의 가장 기본적인 요소를 객체의 단위인 클래스(class)라고 정의하며 그림 1 에서는 클래스의 구조적인 측면의 구성요소를 나타낸다.

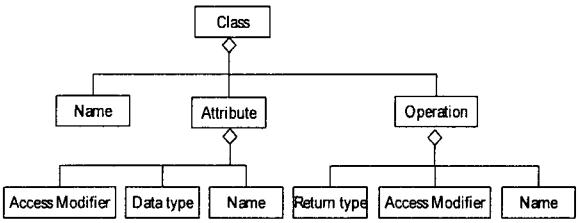


그림 1. 클래스의 구성요소

그림 1은 구조적인 클래스 모델을 나타내고 있고 구성 요소는 플랫폼에 독립적인 요소를 표현하고 있다. 클래스는 이름(Name), 속성(Attribute) 그리고 연산(Operation)의 집합으로 구성되며 이 세가지 주요 구성요소는 클래스를 나타내는 가장 기본적인 단위이다. 따라서 플랫폼 종속적인 모델로 변환 시 가장 중요한 요소가 된다.

4. EJB 어플리케이션을 위한 PSM의 구성요소

상세 설계단계인 PSM 은 PIM 으로부터 변환규칙에 의해서 특정 플랫폼에 맞게 변환된다. EJB 를 위한 PSM 을 표현하기 위해 EJB 의 구성요소를 그림 2 와 같이 표현한다.

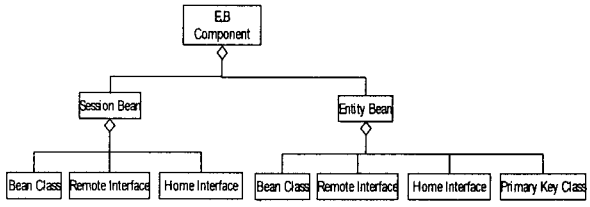


그림 2. EJB 의 구성요소

EJB 에서 엔티티 빈(Entity Beans)의 구성 요소는 빈 클래스(Bean Class), 리모트 인터페이스(Remote Interface), 홈 인터페이스(Home Interface), 프라이머리 키 클래스(Primary Key Class)로 구성되어 있다.

빈 클래스는 비즈니스 로직을 담당하는 빈의 메소드를 구현하고 리모트 인터페이스는 이러한 빈이 어떤 일을 수행하기 위해서 외부에 제공하는 빈의 비즈니스 메소드를 정의한다. 홈 인터페이스는 새로운 빈을 생성하고, 제거하고, 찾아내는 메소드를 정의한다. 프라이머리 키 클래스는 데이터 베이스로의 포인터를 제공하는 클래스이다.

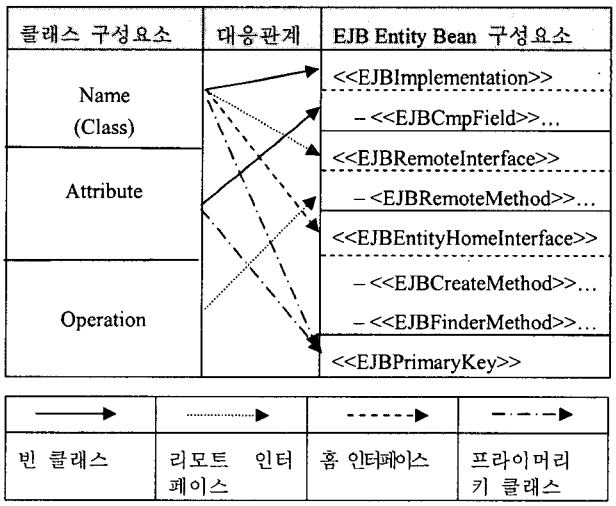
EJB 에서 세션 빈(Session bean)은 비지니스 로직을 포함하고 있는 재사용이 가능한 컴포넌트이며 세션 빈의 상태 유지에 따라 무상태(Stateless)세션과 상태 유지(Stateful)세션 빈으로 나눌 수 있다. 그리고 세션 빈의 구성요소는 빈 클래스와 리모트 인터페이스 그리고 홈 인터페이스로 구성되어 있다. [2]

5. EJB 용 변환 규칙

PIM 의 각 구성요소는 EJB 를 위한 PSM 으로 변환 될 때 EJB 의 각 클래스의 특성에 근거한 변환 규칙에 의해서 변환 된다. 모델간의 효율적인 변환을 위해서는 목표 플랫폼의 특징을 고려하고 이를 대응시킬 수 있는 변환 규칙이 필요하다.

표 1 에서는 PIM 의 구성요소와 EJB 에서 엔티티 빈의 구성요소를 분석 하고 비교함으로써 상호 대응관계를 나타낸 것이다.

표 1. PIM 의 구성요소와 EJB 의 Entity Bean 구성요소와의 대응 관계



PIM 의 클래스로부터 빈 클래스, 리모트 인터페이스, 홈 인터페이스, 프라이머리 키 클래스를 생성하며 생성된 클래스는 PIM 클래스의 이름과 해당 프로파일로 클래스 이름을 생성한다.

빈 클래스의 경우 PIM 의 클래스에서 속성과 연산은 CMP(Container-Managemented Persistence)형태의 속성과 해당 메소드로 변환된다. 리모트 인터페이스는 외부에서 사용할 빈 클래스의 메소드를 정의하고 이 메소드는 빈 클래스의 메소드와 일치해야 한다. 또한 빈 클래스의 속성에 접근하기 위한 접근 메소드(Getter Method, Setter Method)를 포함한다. 홈 인터페이스는 데이터베이스 테이블에 새로운 정보를 추가 하거나 삭제하거나 조회하는 메소드를 생성한다. PIM

에서 해당 클래스를 식별할 수 있는 속성(Key Attribute)을 프라이머리 클래스의 속성으로 변환하고 프라이머리 키를 생성하는 메소드와 키가 유일한지 확인하는 메소드를 생성한다. 만약 PIM 의 클래스에서 해당 식별할 수 있는 속성이 없다면 프라이머리 키를 생성하는 메소드를 통해서 생성된다.

표 1 과 같이 PIM 의 구성요소 및 EJB 를 위한 PSM 의 구성요소의 분석과 각 모델간의 대응되는 관계를 근거로 본 논문에서는 다음과 같이 변환규칙을 정의 한다.

1. PIM의 클래스로부터 빈 클래스, 리모트 인터페이스, 홈 인터페이스 그리고 프라이머리 키 클래스를 각각 생성한다.

   1.1 PIM의 클래스 이름은 빈 클래스는 xxxBean, 리모트 인터페이스는 xxx, 홈 인터페이스는 xxxHome, 그리고 프라이머리 키 클래스는 xxxKey로 각각 클래스 이름을 생성 한다.

   1.2 리모트 인터페이스는 외부에서 사용할 빈 클래스의 메소드를 정의하고 빈 클래스의 속성에 접근하는 게터 메소드(Getter Mothod)와 세터 메소드(Setter)를 생성한다.

   1.3 홈 인터페이스는 테이블의 생성, 삭제 그리고 조회하기 위한 메소드를 각각 생성한다.

   1.4 프라이머리 키 클래스는 데이터 베이스의 포인터를 위한 키를 생성하는 xxxKey() 메소드와 생성된 키가 유일한 키인지 판단하는 equal() 메소드를 생성한다.

2. PIM의 클래스 속성(Attribute)는 대응되는 EJB 빈 클래스의 EJB 속성으로 변환한다.

   2.1 PIM 클래스 속성의 접근 제한자(Access Modifier)와 자료형(Data Type)은 빈 클래스 속성의 접근 제한자와 자료형으로 변환 한다.

3. PIM의 클래스 연산(Operation)은 생성된 EJB 빈 클래스의 메소드로 변환한다.

   3.1 PIM 클래스 연산의 접근 제한자와 자료형은 빈 클래스 연산의 접근 제한자와 자료형으로 변환 한다.

   3.2 빈 클래스의 라이프 사이클을 담당하는 ejbActivate(), ejbPassivate(), ejbRemove(), setEntityContext(), ejbLoad(), ejbStore(), and unSetEntityContext() 메소드를 생성한다.

4. PIM 클래스를 식별하는 속성은 프라이머리 키 클래스의 속성으로 변환된다.

   4.1 equal() 메소드를 통해서 변환된 속성이 유일한 속성이 아니라면 classNameKey() 메소드를 이용해서 속성을 생성한다.

   4.2 PIM 클래스를 식별하는 속성이 없다면 키를 생성하는 메소드를 통해 속성을 생성한다.

이러한 변환 규칙은 EJB 엔티티 빈에서 각 클래스의 특징과 PIM의 클래스의 구성요소를 비교 분석한 결과이다. 그림 3은 비디오 관리 시스템을 이용하여 본 논문에서 정의한 변환 규칙에 따라 PIM에서 EJB를 위한 PSM으로 변환한 것이다.

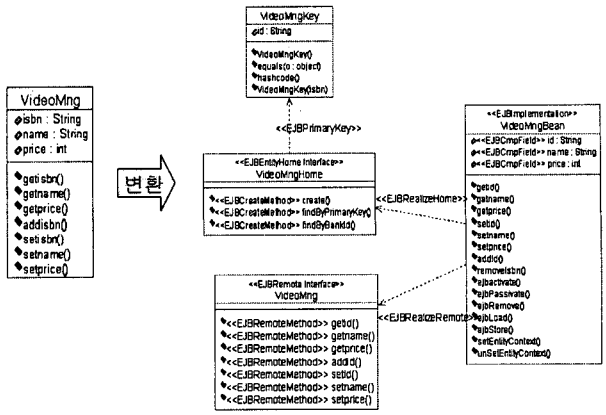


그림 3. 변환 규칙을 적용한 비디오 관리 시스템

본 장에서 정의한 변환 규칙을 이용하여 다양한 플랫폼 환경에서도 적용 가능한 XML같은 템플릿 언어로 표현한다면 기존의 MDA 도구뿐만 아니라 여러 상용 모델링 도구에도 이 변환 규칙을 사용 가능할 수 있게 된다.

6. 결론 및 향후 연구

본 논문에서는 PIM에서의 구성요소와 EJB를 위한 PSM의 구성요소를 체계적으로 비교, 분석함으로써 두 모델의 구성요소간의 대응관계를 통해서 변환 규칙을 정의하였다.

EJB 어플리케이션 개발을 위해 제안된 변환 규칙을 변환 기술에 적용한다면 모델간의 대응관계를 효율적으로 표현할 수 있고 EJB 플랫폼으로의 변환이 용이하며 모델간의 일관성과 추적성을 높일 수 있다. 따라서 제품의 생산성, 유지보수성을 높일 수 있다.

향후 연구로는 다양한 사례분석을 통한 객관적인 산출물을 통해 본 논문에서 정의한 변환 규칙의 검증 및 정제가 필요하다. 그리고 PIM단계의 클래스들간의 관계에서 행위(behavior)적인 요소를 추가 하여 EJB에 대응되는 관계 및 변환 규칙의 정의를 추가해야 한다.

7. 참고문헌

[1] OMG, MDA Guide Version 1.0.1, omg/2003-06-01, June 2003.

[2] E. Roman, *Mastering Enterprise Java Beans*, John Wily & Sons, 1999.

[3] JCP, UML Profile for EJB_Draft, Java Community Process. 2001.

[4] Jos Warmer, Anneke Kleppe, Wim Bast, *MDA Explained*, Addison-Wesley, 2003