컴포넌트 기반의 MDA 공학 프로세스

유태권[0], 라현정, 김수동
숭실대학교 대학원 컴퓨터학과
{tkyu[0], hjla}@otlab.ssu.ac.kr, sdkim@ssu.ac.kr

Component-Based MDA Engineering Process

Tae Kwon Yu[0], Hyun Jung La, Soo Dong Kim
Dept. of Computer Science, Soongsil University

요 약

컴포넌트는 컴포넌트 기반 개발(Component Based Development, CBD)기술에서 재사용되는 기본 단위로서 OOP(Object Oriented Programming, OOP)의 객체보다 상대적으로 큰 단위의 기능성을 제공하며 재사용성이 뛰어나다. 모델 기반 아키텍처(Model Driven Architecture, MDA)는 모델들 간의 다양한 변환(Transformation)을 활용하여 어플리케이션 개발을 자동화하는 새로운 패러다임으로서 소프트웨어의 생산성을 향상시킨다. 그러나 MDA 는 컴포넌트와 같은 어플리케이션들 간의 공통성과 가변성을 활용한 재사용성을 직접적으로 지원하지는 않는다. 본 논문에서는 CBD 와 MDA 의 장점과 한계점을 알아보고 서로의 단점을 보완할 수 있도록 컴포넌트 PIM 과 컴포넌트 기반의 MDA 공학 프로세스를 제안한다. 제안된 컴포넌트 PIM 과 프로세스를 이용하면 어플리케이션들 간의 공통성과 가변성을 이용한 뛰어난 재사용성과 자동화를 통한 생산성의 향상 및 높은 유지보수성을 가질 수 있다.

1. 서론

컴포넌트는 CBD기술에서 재사용되는 기본 단위로서 OOP의 객체보다 상대적으로 큰 기능성을 제공한다. 또한 컴포넌트는 커스트마이제이션 기법을 활용하여 어플리케이션들 간의 공통성과 가변성(Variability)을 지원하므로 OOP의 객체에 비해 재사용성이 뛰어나다. 그러나 컴포넌트는 플랫폼에 종속적이므로 다양한 플랫폼에서 사용하는 경우에 생산성이 저하된다.

MDA는 다양한 모델들 간의 변환을 활용하는 새로운 개발 패러다임이다. MDA는 일련의 변환 규칙에 따라 모델 간 그리고 모델과 실행 가능한 코드 간의 변환을 자동화한다. 따라서, MDA는 어플리케이션 개발을 자동화하여 생산성을 향상 시킨다. 그러나 MDA는 컴포넌트와 같은 어플리케이션들 간의 공통성과 가변성을 이용한 재사용성을 직접적으로 지원하지는 않는다

본 논문에서는 CBD와 MDA의 장점과 한계를 확인한다. 그리고 컴포넌트와 MDA의 장점을 결합시켜 서로의 단점을 보완한 컴포넌트 PIM과 컴포넌트 기반의 MDA 공학 프로세스를 제안한다. 제안한 프로세스를 통해 더 효과적인 어플리케이션 개발을 가능하게 하고, 유지 보수를 용이하게 할 수 있다.

2. 기반 연구

2.1. 컴포넌트

컴포넌트는 여러 다른 어플리케이션에서 재활용을 목표로 하는 CBD의 재사용 단위이다[1]. 따라서, 컴포넌트는 어플리케이션들 간의 공통적인 기능 즉, 공통성을 제공해야 하며 재사용성과 적용성을 향상시키기 위해서 가변성을 제공 해야 한다. 가변성은 가변점(Variation Point)과 가변치(Variant)로 구성 된다. 컴포넌트는 가변성을 설정하기 위한 Customize 인터페이스를 제공한다. 이러한 인터페이스는 구현과 분리되어 있다.

컴포넌트는 가변성을 설정한 후, 조립을 통해 어플리케이션을 개발하므로 재사용성이 뛰어나다. 그러나 컴포넌트는 플랫폼에 종속적이다. 그러므로 기존 컴포넌트의 플랫폼이 아닌 타 플랫폼에서 컴포넌트를 사용하기 위해서는 컴포넌트를 개발하는 중복된 노력이 필요하게 되며 이는 생산성의 저하를 가져온다.

2.2. MDA

MDA는 모델에서 코드 생성 과정을 자동화하여 기존의 어플리케이션 개발 기간을 단축시키는 모델 중심의 새로운 어플리케이션 개발 기술이다[2]. MDA에서는 플랫폼에 독립적인 모델(Platform Independent Model, PIM)을 설계 한 후, 이 PIM을 일련의 변환 규칙을 통해 목표 플랫폼에 종속적인 모델(Platform Specific Model, PSM)로 변환한다. 이 과정은 직접 개발자가 변환하거나 또는 자동화 도구를 사용하여 변환이 가능하다. PSM은 일련의 변환 규칙을 통해 실행 가능한 코드(Code)로 변환된다.

MDA는 PIM에서 코드까지의 과정을 자동화하여 코딩 기간을 단축하여 개발 생산성을 향상시킨다. 또한 PIM의 수정 만으로 실행 코드가 자동으로 수정이 가능하여 유지보수성이 향상한다. 그러나 PIM은 플랫폼에 독립적으로 설계된 모델이지만 컴포넌트의 단위, 인터페이스, 커스트마이제이션을 위한 가변성등의 CBD 기술을 고려하지 않는다[2]. 즉, MDA는 어플리케이션들 간의 공통성과 가변성을 활용한 재사용성을 직접적으로 지원하지 않는다.

3. 컴포넌트 PIM 명세 기법

본 장에서는 CBD와 MDA의 단점을 보완하고 컴포넌트의 재사용성과 MDA의 생산성을 모두 활용하기 위해 컴포넌트 PIM을 제안한다. 컴포넌트 PIM은 플랫폼 독립적이고 조립이 가능한 컴포넌트 단위의 모델이다. 또한, 컴포넌트 PIM은 본 논문에서 제시하는 프로세스의 재사용을 위한 기본 단위가 된다.

컴포넌트 PIM은 기존의 컴포넌트가 바이너리 코드 형태로 재사용 되는 것과 달리 모델 형태로 재사용 된다. 이에 따른 컴포넌트와 컴포넌트PIM의 차이점은 표 1과 같다.

표 1. 컴포넌트와 컴포넌트 PIM의 차이점

기준		컴포넌트	컴포넌트PIM
재사용 형태		바이너리 코드	모델
인터페이스	Provided	O	O
	Required	O	O
	Customize	O	X
커스트마이제이션 시기		배포 단계	PIM설계 단계

컴포넌트 PIM의 인터페이스는 Provided 인터페이스와 Required 인터페이스로 구성되며 클래스들에서 선언된 오퍼레이션들로 구현된다. Provided 인터페이스는 컴포넌트가 다른 컴포넌트 PIM에게 제공하는 기능을 표현한다. Required 인터페이스는 컴포넌트 PIM이 필요로 하는 다른 컴포넌트 PIM의 인터페이스를 표현한다. 또한 각각의 인터페이스는 제약사항을 가지고 있다. 모델 설계 단계에서 커스트마이제이션이 이루어지므로 가변성의 가변치를 설정하기 위한 별도의 Customize인터페이스는 제공하지 않는다.

컴포넌트 PIM의 가변성은 가변성이 발생하는 지점을 나타내는 가변점과 실제로 가변성의 값을 채우는 가변치로 구성된다. 가변성의 Scope는 이진(Binary) Scope, 선택(Selection) Scope, Unknown Scope의 3가지로 나눈다[3]. 이진 Scope와 선택 Scope는 해당 가변성을 가지고 있는 컴포넌트 PIM에서 여러 가변치로서 표현할 수 있고 Unkown Scope는 컴포넌트 PIM의 외부 모델로 표현할 수 있다[4].

컴포넌트 PIM의 설계는 UML 2.0을 기반으로 하되 UML 2.0에 없는 표기 방법은 UML을 확장하여 나타낸다. 클래스, 오퍼레이션 그리고 연관성은 기본 UML2.0을 사용하여 표현하고 인터페이스, 가변점, 가변치는 UML의 확장기능을 사용하여 스테레오타입으로 표현하거나 아이콘으로 표현하며 인터페이스 제약사항은 OCL을 사용하여 표현할 수 있다[5].

4. 프로세스

본 장에서는 컴포넌트 기반의 MDA공학 프로세스와 프로세스를 구성하는 주요한 활동들을 제시한다. 제시되는 프로세스는 컴포넌트 PIM을 생성, 수집하여 조립하고 이를 실행 가능한 코드로 자동화하는 절차를 따른다. 그림 1과 같이 이 프로세스는 총 6개의 활동으로 구성된다.

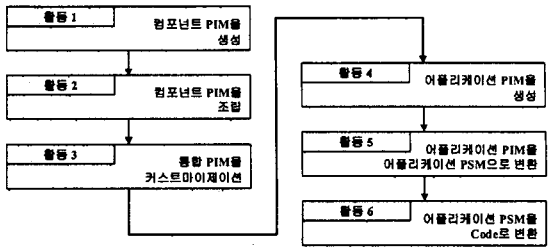

그림 1. 컴포넌트 기반 MDA 공학 프로세스

4.1. 활동 1. 컴포넌트 PIM을 생성

이 활동에서는 기존의 컴포넌트를 위해 설계된 모델을 사용하거나 특정 목적에 맞게 개발된 새로운 컴포넌트를 위해 설계된 모델을 사용하여 조립이 가능하고 플랫폼에 독립적인 컴포넌트 PIM을 생성한다.

PIM 수준에서 커스터마이제이션이 이루어지므로 컴포넌트 PIM은 가변점과 가변치로 구성된 가변성을 스테레오타입을 이용한 UML 확장 모델로 표현할 수 있다. Provide 인터페이스와 Required 인터페이스는 따로 인터페이스 모델을 생성하지 않고 클래스의 메소드를 스테레오 타입을 이용하여 표현할 수 있다. Provided 인터페이스는 ≪Provided Interface≫로, Required 인터페이스는 ≪Required Interface≫로 표현할 수 있다.

도메인을 분석하고 컴포넌트 모델을 설계 하는 과정은 본 연구에 특화된 과정이 아니므로, 일반적인 CDB의 개발 기법과 절차를 사용한다.

4.2. 활동 2. 컴포넌트 PIM을 조립

이 활동에서는 컴포넌트 PIM들을 조립하여 통합 PIM을 생성한다. 컴포넌트 PIM의 조립은 서로 다른 컴포넌트 PIM들 간의 Provided 인터페이스와 Required 인터페이스를 연결하여 이루어진다.

결합하려는 서로 다른 컴포넌트 PIM들 간의 인터페이스 사이에서 충돌이 발생할 경우 직접 모델을 수정하거나 인터페이스들 간의 커넥터를 이용한 매핑 작업으로 해결한다[6].

4.3. 활동 3. 통합 PIM을 커스트마이제이션

이 활동에서는 이전 단계의 산출물인 통합 PIM을 이용하여 컴포넌트를 커스트마이제이션 한다. 커스트마이제이션 과정은 모델과 모델 사이의 수평적 변환 기법(PIM to PIM)을 사용하여 이루어진다[2]. 산출물로는 커스트마이제이션 된 통합 PIM이 생성된다.

일반적인 컴포넌트 커스트마이제이션은 Customize 인터페이스를 통해 배포 단계에서 이루어진다. 그러나 컴포넌트를 PIM으로 표현하여 MDA기술을 이용하면 모델 수준에서 커스트마이제이션이 가능하다. 그러므로 컴포넌트 PIM은 Customize 인터페이스를 제공하지 않는다.

본 논문에서 제시한 커스트마이제이션은 컴포넌트 PIM의 가변치를 직접 설정하여 변환하거나 자동화를 위한 기

법과 도구를 이용할 수 있다. 통합 PIM을 커스트마이제이션 하여 가변성을 설정한 후 Code로 변환하면 Custmize 인터페이스를 포함한 불필요한 Code의 생성을 최소화할 수 있다.

커스트마이제이션 전의 컴포넌트 PIM은 가변점과 가변치가 제공되고 가변성의 종류에 따라서 여러 형태로 표현되지만 커스트마이제이션 후에는 가변점이 없어지고 가변치만 남는다. 그림 2의 예제는 커스트마이제이션 전과 후에 따른 컴포넌트 PIM의 변화를 보여주고 있다.

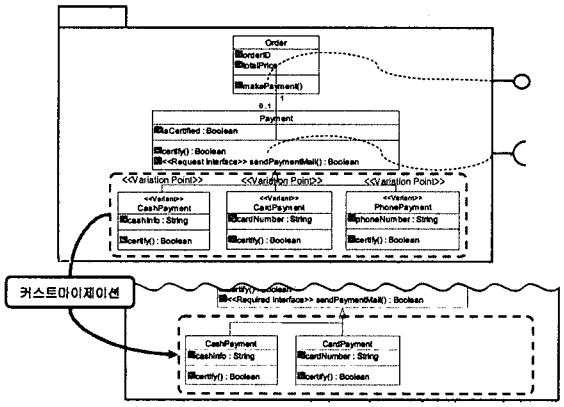

그림 2. 컴포넌트 PIM 의 커스트마이제이션 예제

커스트마이제이션 전에는 Payment 객체를 상속 받는 객체가 3개 존재한다. 그리고 각각의 객체는 각각 이진 Scope의 가변치로서 커스터마이제이션 된 이후에는 CardPayment 객체와 CashPayment 만이 존재한다. makePayment() 메소드는 Provided 인터페이스 로 그리고 sendPaymentMail() 메소드는 Required 인터페이스로 제공된다.

4.4. 활동 4. 어플리케이션 PIM 을 생성

이 활동의 첫번째 단계에서는 이전 단계의 산출물인 통합PIM에서 나타내지 못한 어플리케이션 특정 정보를 PIM으로 구현한다.

두번째 단계로는 커스트마이제이션 된 통합 PIM에 모델과 모델 사이의 통합(Integrate) 기법을 사용하여 통합 PIM과 어플리케이션 특정 PIM을 통합한 어플리케이션 PIM을 생성한다.

4.5. 활동 5. 어플리케이션 PIM 을 어플리케이션 PSM 으로 변환

이 활동에서는 어플리케이션 PIM을 변환 규칙, 템플릿 등을 이용하여 어플리케이션 PSM을 자동 생성하거나 직접 변환하여 생성할 수 있다.

PSM을 생성 하기 위한 변환 규칙에는 PIM에서 사용자 정의 타입을 이용해 표현한 인터페이스와 제약사항에 관한 변환 규칙의 정의가 필요하다. 각 인터페이스는 PSM에서도 인터페이스로서 표현이 되어야 한다.

4.6. 활동 6. 어플리케이션 PSM 을 Code 로 변환

이 활동에서는 이전 단계에서 생성된 어플리케이션 PSM 모델을 기반으로 실행 코드를 생성한다. 이 변환은 개발자가 직접 변환하거나, 자동화 도구를 이용하여 변환할 수 있다.

활동 5와 6은 본 연구에 특화된 과정이 아니므로, 일반적인 MDA의 변환 기법과 절차를 사용한다.

5. 결론 및 향후 연구과제

컴포넌트는 CBD기술에서 재사용되는 기본 단위이다. 컴포넌트는 여러 다른 어플리케이션에서 사용하기 위해 공통성과 가변성을 포함하고 있으므로 재사용성과 생산성이 뛰어나다. MDA는 플랫폼에 독립적인 PIM을 일련의 변환 규칙에 따라 PSM과 실행 코드를 자동으로 생산하는 소프트웨어 자동화 기술이다.

본 논문에서는 실용적이고 효율적으로 적용 가능한 MDA 공학의 새로운 시도로서 컴포넌트 기반의 MDA공학 프로세스를 제안하였다. 제안한 프로세스에서는 먼저 컴포넌트를 컴포넌트 PIM으로 표현하고 이를 통합 PIM으로 조립하여 각 어플리케이션의 요구사항에 맞게 커스트마이제이션 한 후, 어플리케이션 특정 정보를 추가로 통합한 어플리케이션 PIM을 생성한다. 생성된 어플리케이션 PIM은 일련의 규칙에 따라 어플리케이션 PSM으로 변환된다. 어플리케이션 PSM은 Code로 변환 된다. 모든 변환은 직접 변환하거나 자동화 도구를 이용하여 변환할 수 있다. CBD와 MDA 두 기술의 장점을 갖춘 제안된 프로세스를 이용하여 재사용 기본 단위를 컴포넌트 PIM으로 개발하면 보다 효과적인 어플리케이션 개발과 높은 재사용성, 유지보수성이 기대된다.

앞으로 향후 연구에서는 컴포넌트 PIM에 대한 더욱 상세한 정의와 컴포넌트 PIM 모델링의 지침을 제안하고 프로세스의 지침을 상세화하며 모델들 간의 변환 기법을 정의 해야 할 것이다.

6. 참고문헌

[1] Heineman, G. and Councill, W., *Component-Based Software Engineering*, Addison Wesley, 2001.

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

[3] 소동섭, 김수동, "컴포넌트 가변성 유형 및 Scope 에 대한 정형적 모델," 한국정보과학회, 제 30 권, 제 4 호, 2003.

[4] Kim S., Min H., and Rhew S., "Variability Design and Customization Mechanisms for COTS Components ," LNCS 3480, *Proceedings of The 2005 International Conference on Computational Science and its Applications (ICCSA 2005)*, 2005.

[5] Frankel, D., *Model Driven Architecture™:Applying MDA™ to Enterprise Computing*, Wiley, 2003.

[6] Min H, Choi S, and Kim S, "Using Connectors to Resolve Partial Matching Problems in COTS Component Acquisition ," LNCS 3054, *Proceedings of 7th International Symposium on Component-based Software Engineering (CBSE 7)*, 2004.