ACAB : 소프트웨어 프로덕트 라인 지원을 위한 컴포넌트 개발 도구

정주미[O] 최승훈
덕성여자대학교 전산정보통신학과
{ jumi[O], csh }@duksung.ac.kr

ACAB : Component Asset Builder for supporting Software Product Lines

Ju-mi Jung[O] Seung-Hoon Choi
Dept. of Computer Science, Duksung Women' s University

요 약

최근 컴포넌트 기반의 소프트웨어 프로덕트 라인에 대한 연구와 소프트웨어 프로덕트 라인에 자동 생성 프로그래밍 기법을 적용하기 위한 연구가 활발히 진행 중이다. 본 논문에서는 재사용 가능한 컴포넌트 자산을 구축하고, 재구성 자동화를 통해 컴포넌트 코드를 생성하는 시스템을 제안한다. 본 도구는 컴포넌트 기반 프로덕트 라인 개발 방법론과 자동 생성 프로그래밍 기법, XML/XSLT 기술을 이용하여 구축되었다. Component Asset 개발자 레벨에서 사용하는 컴포넌트 자산 구축기와 재사용자 레벨에서 사용하는 컴포넌트 코드 생성기로 구성되어 있으며, 컴포넌트 기반의 소프트웨어 프로덕트 라인 개발에 효과적으로 활용될 수 있다.

1. 서론

소프트웨어 프로덕트 라인이란, 공통된 핵심 소프트웨어 자산들로부터 특정 목표나 시장을 위해 개발된 소프트웨어 집약 시스템들의 집합을 의미한다. 소프트웨어 컴포넌트는 여러 소프트웨어 시스템으로 조립될 수 있는 기본 block[1]이며, 코드 중복을 최소화하고 재사용을 최대화한다.

본 논문은 재사용 가능한 컴포넌트 자산 구축기와 재구성 자동화를 이용한 컴포넌트 코드 생성 시스템을 제안한다. 효율적인 프로덕트 라인 구축 및 구체적인 제품 생산을 위해, 컴포넌트 기반 프로덕트 라인 개발 방법론인 PLUS[2]와 자동 생성 프로그래밍의 한 기법인 GenVoca[3]의 계층 구조, 그리고 XML 기반의 컴포넌트 자동 생성 기법의 결합을 이용한다.

본 논문의 구성은 다음과 같다. 2장에서 ACAB 도구의 구성을 소개하고, Microwave Oven 프로덕트 라인을 예제로 하여 3장에서 컴포넌트 자산 구축기의 구조를, 4장에서 컴포넌트 코드 생성기의 구조를 설명한다.

2. Adaptable Component Asset Builder (ACAB)

ACAB는 Component Asset 개발자 레벨에서 사용하는 컴포넌트 자산 구축기와 재사용자 레벨에서 사용하는 컴포넌트 코드 생성기로 구성되어 있으며, 전체 구조는 그림 1과 같다. Component Asset 개발자는 특성 모델과 컴포넌트 계층 구조를 정의하고, 조립 규칙과 코드 템플릿을 작성한다. 재사용자는 특정한 컴포넌트의 요구사항을 입력하여 특성 모델로부터 특성 구성을 만들고, 자동 생성된 코드를 얻는다.

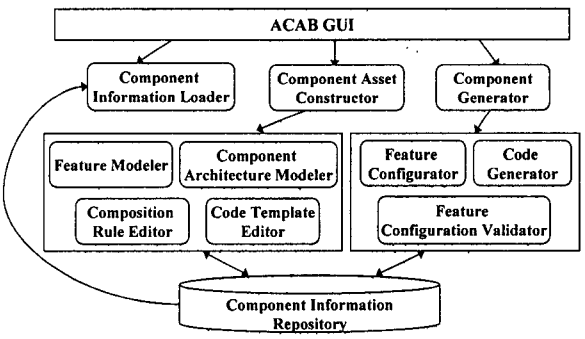


그림 1 ACAB Architecture

그림 2는 본 도구 구현에 사용된 컴포넌트 모델을 보여준다. 컴포넌트의 종류로는 Leaf와 Composite가 있다. Interface와 구현 부품으로 정의되는 계층들은 상속 또는 집합 관계로 조립된다. 각 인터페이스와 구현 부품을 위한 XSLT 코드 템플릿은 클래스 코드로 변환된다.

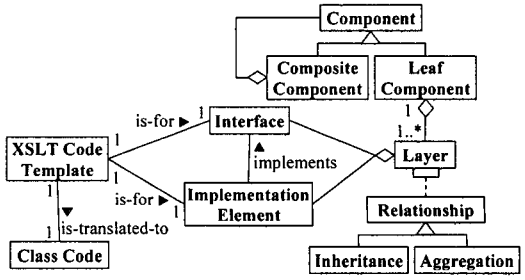


그림 2 ACAB에서의 컴포넌트 모델

3. Component Asset Constructor의 구성

본 논문은 컴포넌트 기반 프로덕트 라인 개발 방법론의 하나인 PLUS[2]를 적용한 MicrowaveOven 프러덕트 라인을 예제로 하여 도구의 기능을 기술한다. MicrowaveOven의 소프트웨어 아키텍처는 그림 3과 같다. 그림 3에서 생략된 《output component》로는 DisplayInterface와 DisplayPrompts로 구성된 복합 컴포넌트 MicrowaveDisplay가 있으며, 제어를 담당하는 MicrowaveControl은 3개의 서브 컴포넌트(MicrowaveOvenControl, OvenTimer, OvenData)를 포함한다.

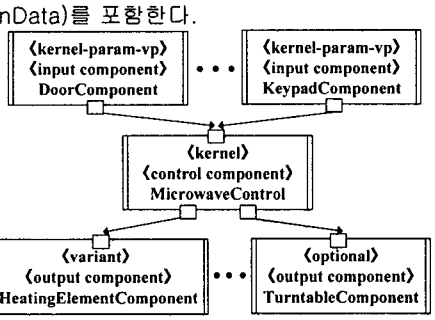


그림 3 MicrowaveOven의 구성 컴포넌트

3.1 Feature Modeler

특성은 구체적인 컴포넌트의 기능과 형태에 직접적인 영향을 주는 중요한 특징들을 나타낸다. 도메인 분석을 통해 컴포넌트 패밀리가 가지는 공통성은 의무적 특성으로, 가변성은 선택적 또는 택일적 특성으로 분류하고 이를 바탕으로 특성 모델을 작성한다.

특성 모델에서 선택된 특성에 따라 어떤 컴포넌트가 영향을 받는지를 분석하여 특성과 컴포넌트들 사이의 의존성을 식별한다. 그림 4는 MicrowaveOvenControl과 HeatingElementComponent에 영향을 미치는 특성들로만 이루어진 특성 모델을 보여준다.

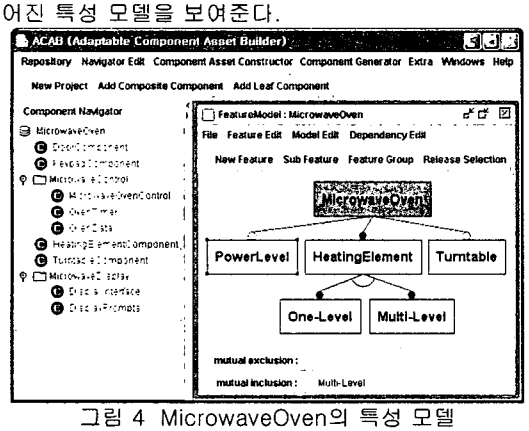


그림 4 MicrowaveOven의 특성 모델

3.2 Component Architecture Modeler

GenVoca[3]에서 제안한 계층 구조를 이용한 자동 생성 프로그래밍 기법을 이용하여 세 단계를 통해서 각 컴포넌트의 계층 구조를 정의한다. 첫째, 컴포넌트 카테고리와 각 카테고리별 구현 컴포넌트들을 식별한다. 둘째, 컴포넌트 카테고리 간의 계층 관계를 설정한다. 셋째, 각 계층별 인터페이스를 정의한다. 컴포넌트 카테고리는 표준화된 인터페이스를 제공하는 하나의 계층이며 각 계층은 필수적 계층과 선택적 계층으로 분류된다. 계층들은 상속 또는 집합 관계로 조립되며 그림 5에서 상속 관계는 실선으로, 집합 관계는 점선으로 표현되어 있다.

예로 보여진 IMicrowaveOvenControl 계층은 IBasicMicrowaveOvenControl 계층을 상속하며, MicrowaveOvenControl을 구현 부품으로 가진다. MicrowaveOvenControl은 인터페이스에서 정의한 메소드 sendControlRequest(ControlRequest newCr)를 구현하게 된다.

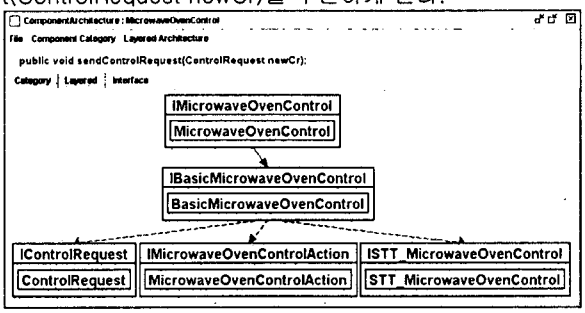
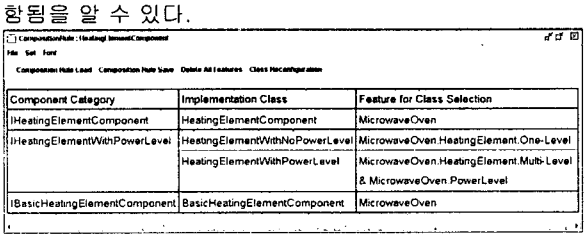


그림 5 MicrowaveOvenControl의 계층 구조

3.3 Component Composition Rule Editor

조립 규칙은 재사용자에 의해 선택된 특성에 따라 컴포넌트 아키텍처를 이루는 각 계층별로 어떤 구현 부품을 구체적인 컴포넌트에 포함시킬 것인가를 결정한다.

그림 6을 보면, IHeatingElementWithPowerLevel 카테고리의 구현 클래스 중에서 HeatingElementWithPowerLevel은 재사용자에 의해 Multi-Level 특성과 Power Level 특성이 모두 선택되면 구체적인 코드 생성 시에 포함됨을 알 수 있다.



Component Category	Implementation Class	Feature for Class Selection
IHeatingElementComponent	HeatingElementComponent	MicrowaveOven
IHeatingElementWithPowerLevel	HeatingElementWithNoPowerLevel	MicrowaveOven.HeatingElement.One-Level
	HeatingElementWithPowerLevel	MicrowaveOven.HeatingElement.Multi-Level & MicrowaveOven.PowerLevel
IBasicHeatingElementComponent	BasicHeatingElementComponent	MicrowaveOven

그림 6 HeatingElementComponent의 조립 규칙

3.4 Code Template Editor

XSLT 스타일시트는 구체적인 컴포넌트 코드 생성시 필요한 코드 템플릿을 정의하는데 이용된다.

각 컴포넌트에 대한 파일 목록에서 작성할 파일을 선택하면 Component Architecture Modeler(3.2절 참조)의 세 번째 단계에서 입력한 메소드와 기본적인 코드 골격이 자동으로 입력된다. 또한, 재사용자가 제공한 커스터마이징 데이터를 바탕으로 특정 구현 컴포넌트와 관련된 코드 부분을 선택하는 기능을 제공한다.

그림 7과 같이 각 컴포넌트의 모든 인터페이스와 구현 클래스 별로 코드 템플릿을 작성하면 컴포넌트 자산 구축 과정이 완료된다.

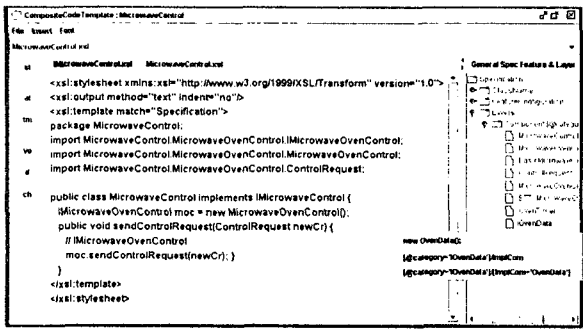


그림 7 MicrowaveControl의 코드 템플릿

4. Component Generator의 구성

4.1 Feature Configurator

특성 구성[4]이란 특정한 컴포넌트를 구성하기 위해 재사용자가 입력한 특성 모델에 대한 요구 사항이다. 재사용자는 컴포넌트 패밀리 내에 존재하는 가변성을 나타내는 선택적 특성의 포함 여부를 결정하며 택일적 특성들 중 하나를 선택하고, 논리합(OR) 관계의 특성들을 선택하거나 인자화 특성에 대한 값을 입력한다.

특성 구성을 저장할 때는 상호 의존적이거나 상호 배타적인 특성 의존성 관계에 따라 특성 선택이 올바르게 되었는지 확인한다.

그림 8은 특성 구성의 한 예를 보여준다. 선택적 특성인 Turntable 특성과 HeatingElement 특성의 택일적 특성들 중 One-Level 특성이 재사용자에 의해 선택되었다.

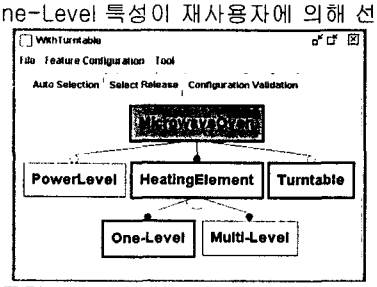


그림 8 MicrowaveOven의 특성 구성

4.2 Feature Configuration Validator

특성 구성 검증은 특성 선택에 따라 각 계층 별로 구현 클래스가 선택되어지는지 여부와 조립 규칙에 따라 필수적인 계층들이 포함되었는지를 확인한다.

재사용자가 코드를 생성하고 싶은 컴포넌트 패밀리를 선택하면, 특성 구성이 검증된 경우 XML 형태의 내부 명세서가 자동으로 생성된다. 내부 명세서는 커스터마이징된 특성의 목록과 부품 생성에 필요한 계층 및 각 인터페이스의 구현 클래스 목록으로 구성된다.

4.3 Code Generator

재구성된 컴포넌트 코드를 자동 생성하는 Code Generator의 주된 기능은 2가지이다.

첫번째 기능은 XSLT 프로세서를 이용한 코드 생성 기능이다. 내부 명세서에 Code Template Editor에서 작성한 코드 템플릿을 적용하여 컴포넌트 코드를 자동으로 생성한다. 그림 9는 ACAB를 사용하여 자동 생성한 코드 중 하나인 HeatingElementWithNoPowerLevel.java 파일이다.

두 번째 기능은 생성된 코드를 읽거나 수정한 후에 compile 하고 실행하는 기능이다. 이 과정을 통해 생성된 코드의 오류 정정 및 테스트가 가능하다.

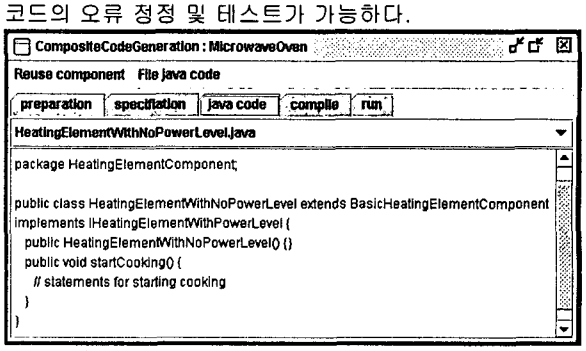


그림 9 HeatingElementWithNoPowerLevel.java

5. 결론 및 향후 연구 과제

본 논문에서는 도메인 공학의 주요 산물인 특성 모델로부터 특정한 컴포넌트의 요구사항을 입력받아 특성 구성을 만들고, 컴포넌트 명세서와 XSLT 코드 템플릿을 이용하여 재구성된 컴포넌트 코드를 자동 생성하는 도구를 구현하였다.

본 도구는, 소프트웨어 프러덕트 라인 아키텍처를 구성하는 컴포넌트의 종류 선택 및 컴포넌트 내부의 재구성을 지원한다. 또한, 복합 컴포넌트 또는 leaf 컴포넌트를 조합하여 상위의 복합 컴포넌트 조립을 가능하게 하여 더 큰 단위의 컴포넌트 코드를 자동 생성해 준다.

본 논문의 결과는 컴포넌트 기반 소프트웨어 프로덕트 라인 개발의 생산성을 향상시키는데 활용될 수 있다. 현재, 본 연구 결과를 임베디드 소프트웨어 프러덕트 라인에 적용시키는 방법에 대한 연구가 진행 중이다.

참고문헌

[1] Krzysztof Czarnecki and Ulrich W. Eisenecker, "Generative Programming. Methods, Tools, and Applications ", Addison-Wesley, 2000.

[2] C. Atkinson et al., "Component-based Product Line Engineering with UML" , Addison-Wesley, 2002.

[3] J. Blair and D. Batory, "A Comparison of Generative Approaches : XVCL and GenVoca ", Technical Report December, 2004.

[4] S. Thiel and A. Hein, "Systematic Integration of Variability into Product Line Architecture Design ", SPLC2 2002, LNCS 2379, pp.130-153, 2002, Springer-Verlag.