특성 구성을 이용한 컴포넌트 재구성 자동화

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

Automatic Component Reconfiguration using Feature Configuration

Ju-Mi Jeong[0] Jeong-Ah Jang Seung-Hoon Choi
Dept. of Computer Science, Duksung Women's University

요 약

소프트웨어 개발 단계 초기에 소프트웨어 제품군에 속하는 멤버들의 공통점과 차이점들을 미리 예측하고 분석하여 단위가 크고 전략적인 형태의 재사용을 가능하도록 함으로써 궁극적으로 소프트웨어 개발의 생산성을 향상시키고자 하는 것이 소프트웨어 프로덕트 라인 개발 방법론의 목적이다. 최근에는 소프트웨어 프로덕트 라인 개발에 자동 생성 프로그래밍 기법을 적용하기 위한 연구가 진행 중이다. 그런데, 어플리케이션 단위의 프로덕트 라인 개발 방법론은 단위가 너무 커서 자동 생성 기법을 효율적으로 적용하기가 힘들며, 따라서 그 초점을 컴포넌트 단위로 한정시킬 필요가 있다. 본 논문에서는 도메인 공학의 주요 산물인 특성 다이어그램으로부터 특정 컴포넌트의 요구사항을 나타내는 특성 구성(Feature Configuration)을 만들고, 이를 바탕으로 컴포넌트 코드를 자동 생성하는 도구를 구현하였다. 본 논문의 컴포넌트 재구성 자동화 도구는 재사용자의 요구에 맞는 컴포넌트 소스 코드를 자동 생성함으로써 소프트웨어 프로덕트 라인 개발 생산성을 향상시킨다.

1. 서 론

소프트웨어 개발 단계 초기에 소프트웨어 제품군에 속하는 멤버들의 공통점과 차이점들을 미리 예측하고 분석하여 단위가 크고 전략적인 형태의 재사용을 가능하도록 함으로써 궁극적으로 소프트웨어 개발의 생산성을 향상시키고자 하는 것이 소프트웨어 프로덕트 라인 개발 방법론의 목적이다[1]. 최근에는 소프트웨어 프로덕트 라인 개발에 자동 생성 프로그래밍 기법을 적용하기 위한 연구가 활발히 진행 중이다 [2] [3].

그런데, 어플리케이션 단위의 프로덕트 라인 개발 방법론은 단위가 너무 커서 자동 생성 기법을 효율적으로 적용하기가 힘들며, 따라서 공통점과 차이점을 분석하고 소스 코드를 자동 시키기 위한 방법의 초점을 특정 컴포넌트로 제한시킬 필요가 있다.

본 논문에서는 도메인 공학의 주요 산물인 특성 모델로부터 특정한 컴포넌트의 요구사항을 입력 받아 특성 구성(Feature Configuration)을 만들고 이를 이용하여 컴포넌트 코드를 자동 생성하는 도구를 구현하였다. 본 논문의 컴포넌트 재구성 자동화 도구는 컴포넌트를 구성하는 부품들을 재사용자의 요구에 맞게 재구성하여 소스 코드를 자동 생성함으로써 소프트웨어 개발 생산성을 향상시킬 뿐 만 아니라, 컴포넌트 기반 프로덕트 라인 개발 방법론의 기반 기술로 이용될 수 있다.

본 논문의 전체적인 구성은 다음과 같다. 제 2 장에서 관련 연구를 소개하고, 제 3 장에서 연결리스트 컴포넌트 패밀리를 예로 하여 특성 모델과 특성 구성을 설명한다. 제 4 장에서 컴포넌트 재구성 자동화 도구의 구현을 기술한 후, 제5장에서 결론 및 향후 연구 과제를 기술한다.

본 연구는 한국과학재단 목적기초연구(과제번호 R06-2002-003-01006-0)지원으로 수행되었음.

2. 관련 연구

2.1 도메인 공학과 특성 모델

소프트웨어 프로덕트 라인 개발 방법론에서 어플리케이션간의 공통점과 차이점을 분석하기 위한 영역 분석 방법 중 대표적인 것으로서, 특성 중심의 분석 방법을 이용하여 영역에서의 공통점과 차이점을 분석하는 FODA가 있으며, 이를 확장한 재사용 기법이 FORM이다[4]. 이 기법에서 영역 분석의 주요 결과물인 특성 모델(*Feature Model*)은 제품군에 속하는 멤버들 사이의 공통점과 차이점에 대한 고수준의 추상화를 제공하는 요구사항 모델이다. 특성이라는 것은, 각 소프트웨어 제품들에서 구현되고 테스트되고 배포, 유지되어야 하는 기능적 추상화를 뜻하는 것으로 요구 사항이나 기능들을 의미한다[5].

2.2 자동 생성 프로그래밍

소프트웨어 프로덕트 라인에서 구체적인 시스템을 생산하기 위한 기법으로 여러 가지가 존재한다. 재사용자의 많은 개입이 필요한 조립 방법, 재사용자의 약간의 개입이 필요한 반자동식 조립 방법, 재사용자가 제시한 차이점을 바탕으로 한 자동 생성 방법 등이 그것이다. 특히, 프로그램 자동 생성에 관한 연구가 최근 활발히 진행되어, 프로그램 자동 생성을 위한 여러 가지 분석 방법과 구현 방법이 제안되었다[2].

[3]에서는 문서 표현의 표준 기술인 XML(eXtensible Markup Language)을 이용하여 프로그램의 추상적 명세서를 작성하고, 이를 받아들여 프로그램 코드를 자동 생성하는 프로그램 생성기를 제안하였다. [6]에서는 프로덕트 패밀리를 구성하는 자산들을 가변성을 수용하기 위한 방법으로 구조화하고, XML을 기반으로 한 명세서를 해석하여 자산의 커스터마이징을 반자동화(semi-automatic support)하는 도구에 대한 연구를 수행하였다.

3. 특성 모델과 특성 구성

영역 분석의 결과물인 특성 모델(Feature Model)은 제품군에 속하는 멤버들 사이의 공통점과 차이점에 대한 고수준의 추상화를 제공하는 요구사항 모델이다.

특성 모델에서 제품군에 속하는 멤버들이 공유하는 공통점은 필수적 특성(mandatory feature)으로 표현되며, 멤버들 간의 차이점은 선택적(optional), 택일적(alternative), 비결정(free parameter) 특성 등으로 표현된다. 필수적 특성은 어플리케이션에 반드시 포함되어야 하는 특성을 의미하고, 택일적 특성은 여러 개 중에서 하나 선택되어지는 특성을 의미한다. 선택적 특성은 특정 제품에 포함되거나 또는 포함되지 않는 특성을 의미하며, 비결정 인자 특성은 특정 제품 생산 시 재사용자가 특정 값을 제공해 주어야 하는 특성을 의미한다.

그림 1 의 특성 모델은, 본 논문에서 구현된 특성 모델링 도구를 이용하여 모델링 된 연결리스트 컴포넌트 특성 모델로서, 본 논문의 예제로 사용된다. 사각형은 각각의 특성을 나타내며, 사각형 위의 검은 원은 필수적 특성을, 하얀 원은 선택적 특성을, 원호 아래에 위치한 특성들은 택일적 특성, 각 괄호([ ])로 나타내어진 특성은 비결정 인자 특성을 나타낸다.

그림 1 에서 **ElementType** 특성은, 연결 리스트의 원소 형을 나타내며, **Ownership** 특성은 리스트를 구성하는 원소에 대한 소유권을 의미한다. **Morphology** 특성은 생성된 리스트가 서로 다른 형의 원소를 포함할 수 있는지의 여부을 나타내며, **LengthCounter** 특성은 연결 리스트의 길이를 계산하는 카운터의 포함 여부를 의미하고, **Tracing** 특성은 연결리스트에 대한 각각의 연산에 대한 추적 기능의 포함 여부를 나타낸다.

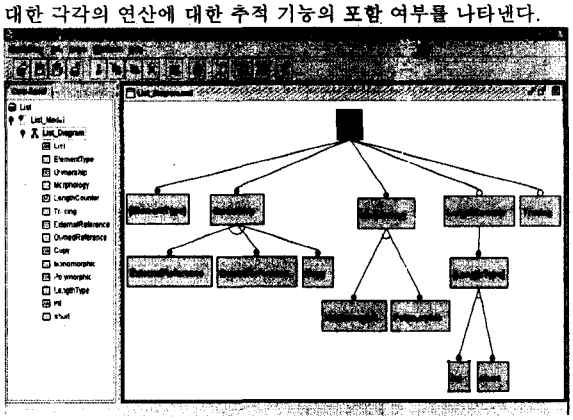

그림 1 연결리스트 컴포넌트의 특성 모델 예제

이 특성 모델에 의하면, 연결 리스트 패밀리에 속하는 멤버들은 **ElementType, Ownership, Morphology** 등의 공통적인 특성을 가지며, **LengthCounter, Tracing** 등의 추가적인 기능을 가질 수 있다. 각 특성의 실제 값은, 재사용자가 하위의 택일적 특성들 중 하나의 값을 선택하거나 직접 값을 입력함으로써 결정된다. 본 논문에서는, 특성 모델이, 컴포넌트 패밀리의 멤버들 사이의 공통점과 차이점을 표현하기 위한 모델 뿐 만 아니라 재사용자가 특정 소프트웨어를 생성할 때 요구 사항을 입력하기 위한 장치로서 사용된다. 즉, 재사용자는 특성 모델로부터 원하는 특성을 선택하거나 비결정 특성의 값을 입력함으로써, 원하는 특정 컴포넌트 멤버에 대한 요구 사항을 표현한다. 특성 모델로부터 재사용자가 결정한 특성 선택 결과를 특정 제품에 대한 "특성 구성(Feature Configuration)" 이라고 한다[7].

4. 컴포넌트 재구성 자동화 도구

4.1 특성 구성 생성

그림 2는 특성 모델로부터 재사용자가 요구사항을 입력하여 특성 구성을 만들고 이를 바탕으로 특정 컴포넌트를 생성할 수 있는 컴포넌트 재구성 자동화 도구를 보여준다.

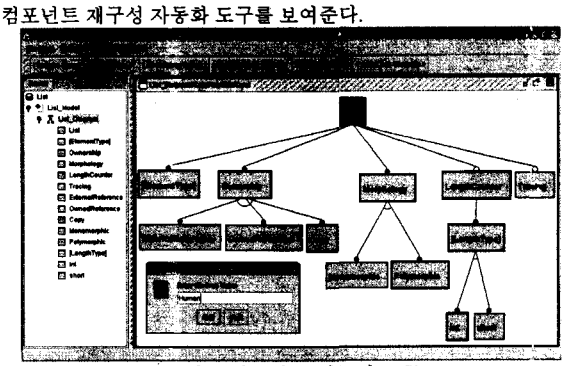

그림 2 컴포넌트 재구성 도구

이 도구에서 재사용자가 특정 컴포넌트에 대한 특성 구성을 만들기 위해서 실행할 수 있는 동작은 다음과 같다.

(i) 비결정인자 특성에 대해서 실제 값을 제공한다.

(ii) 선택적 특성에 대해서 선택 또는 해제를 결정한다.

(iii) 택일적 특성들을 하위 특성으로 가진 특성에 대해서는, 하위의 택일적 특성 중에서 하나를 선택한다.

표 1은, 그림 1 의 특성 모델에 표현되어 있는 특성 중에서 특성 구성에 영향을 미치는 특성들과 선택 결과 및 이로 인해 생성될 특정 컴포넌트를 보여주는 표이다. 표 1 의 특성 구성이 나타내는 ' 연결리스트 1' 컴포넌트는, 리스트의 원소 형은 Human 클래스이고 기존 Human 객체에 대한 복사본을 유지하며, 리스트의 실제 원소는 Human 의 sub 클래스형 모두 가능하며, 현재 원소 개수를 유지하는 카운터를 갖는 연결 리스트를 나타낸다. 그림 2 는, 이 표에 나타낸 특성 구성을 재사용자가 만들어내는 단계를 보여준다.

표 1 그림 1로부터 가능한 특성 구성의 예

특성	종류	특성 구성
[ElementType]	필수적/비결정인자	값 = " Human"
Ownership	필수적/하위택일	Copy
Morphology	필수적/하위택일	Polymorphic
LengthCounter	선택적	yes(선택됨)
[LengthType]	필수적/비결정/하위택일	int
Tracing	선택적	no(선택안됨)
생성되는 구체적인 컴포넌트		연결리스트1

4.2 컴포넌트 소스 코드 자동 생성

재사용자가 생성한 특성 구성으로부터 특정 컴포넌트의 소스 코드를 생성하는 도구의 전체적인 구조는 그림 3과 같다. 특성 구성 결과는 XML 형식의 컴포넌트 구성 명세서로 변환되며, 특성 구성 처리기는 컴포넌트 구성 명세서와 조립 규칙을 입력받

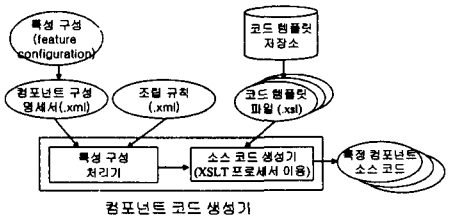

그림 3 컴포넌트 자동 생성 도구의 아키텍처

아 재사용자가 원하는 컴포넌트가 어떤 부품들로 조립되는지를 나타내는 내부 데이터를 소스 코드 생성기에 전달한다. 소스 코드 생성기는 컴포넌트 조립에 필요한 부품들의 코드 템플릿인 XSLT 스크립트 파일들을 처리하여 최종 컴포넌트 소스 코드를 생성한다.

그림 4는 표 1의 특성 구성에 대한 컴포넌트 구성 명세서의 압축된 일부를, 그림 5는 조립 규칙을 나타내는 XML 파일의 압축된 일부를 보여준다.

```
<Specification>
  ... ...
  <Features>
   <Feature><name>List</name></Feature>
   <Feature><name>ElementType</name><value>Human</value></Feature>
   <Feature><name>Ownership</name></Feature>
   <Feature><name>LengthCounter</name></Feature>
   <Feature><name>Copy</name></Feature>
   <Feature><name>Polymorphic</name></Feature>
   <Feature><name>LengthType</name><value>int</value>
   </Feature>
   ... ...
  </Features>
</Specification>
```

그림 4 표1의 특성 구성에 대한 컴포넌트 구성 명세서

```
<Component_category name="Copier">
  <ImplCom name="MonomorphicCopier">
    <OR_cond> <AND_cond>Copy</AND_cond>
              <AND_cond>Monomorphic</AND_cond> </OR_cond>
  </ImplCom>
  <ImplCom name="PolymorphicCopier">
    <OR_cond><AND_cond>Copy</AND_cond>
         <AND_cond>Polymorphic</AND_cond></OR_cond>
  </ImplCom>
  <ImplCom name="EmptyCopier">
    <OR_cond><AND_cond>ExternalReference</AND_cond></OR_cond>
    <OR_cond><AND_cond>OwnedReference</AND_cond></OR_cond>
  </ImplCom>
</Component_category>
```

그림 5 연결리스트 생성을 위한 조립 규칙

그림 6은 표 1의 특성 구성에 의해 재구성된 ' 연결리스트 1' 의 소스 코드의 압축된 일부를 보여준다.

```
public class PtrList implements BasicList {
  ElementDestroyer Destroyer = new ElementDestroyer();
  EmptyTypeChecker TypeChecker = new EmptyTypeChecker();
  PolymorphicCopier Copier = new PolymorphicCopier();
  Human head_ = null;  LengthList tail_;
  public PtrList(Human h) {
    setHead(h);    tail_ = null;      }
  public PtrList(Human h, LengthList t) {
    setHead(h);    setTail(t);        }
  ... ...
}
```

그림 6 표1의 특성 구성에 의해 생성된 자바 코드의 일부

4.3 GenVoca 기법을 이용한 도메인 공학

본 논문에서는 소스 코드의 자동 생성이 가능하도록 하기 위해 [2]에서 제시된 자동 프로그래밍 기법의 하나인 GenVoca 기법 예제로 사용된 연결리스트 컴포넌트의 계층 구조 및 부품을 사용하였다. 그림 7은 연결리스트의 GenVoca 계층 구조를 나타내며, 그림 8은 GenVoca 도메인 분석 통해 식별된 컴포넌트 카테고리를 보여준다.

본 논문에서 구현된 재구성 자동화 도구는, 그림 7에 표현되어 있는 아키텍처로부터 추출된 조립 규칙, 그림 8의 각 부품의 코드 템플릿을 제공하는 XSLT 스크립트, 특성 구성에 의해 만들어진 컴포넌트 구성 명세서 등을 이용하여 재사용자가 원하는 컴포넌트 소스 코드를 자동 생성한다.

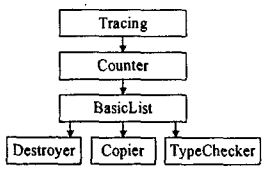

그림 7 연결리스트 컴포넌트의 계층 구조

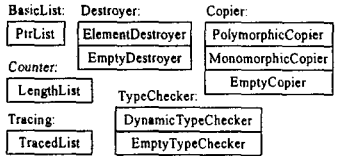

그림 8 연결리스트에 대한 컴포넌트 카테고리

5. 결론 및 향후 연구 과제

현재 어플리케이션 단위의 크고 전략적인 형태의 재사용을 가능하도록 하는 소프트웨어 프로덕트 라인 개발 방법론에 대한 연구가 활발히 진행 중이다. 최근에는 프로덕트 라인의 개발 생산성을 향상시키기 위한 자동 생성 기법에 대한 연구가 진행되고 있다. 그러나, 어플리케이션 전체에 자동 생성 기법을 적용하기에는 어려움이 크며, 따라서 컴포넌트 단위의 자동 생성을 통해 프로덕트 라인 구축의 생산성을 향상시킬 수 있다.

본 논문에서는, 재사용자가 원하는 컴포넌트에 대한 요구 사항을 입력 받는 장치로 특성 모델을 사용하고, 이로부터 만들어진 특성 구성에 따라 재구성된 컴포넌트 소스 코드를 자동으로 생성하는 도구를 구현하였다. 본 논문의 결과는 컴포넌트 패밀리 개발 뿐 만 아니라 프로덕트 라인 개발 생산성을 향상시킬 수 있는 기반 기술로 이용될 수 있을 것이다.

향후 과제로는 보다 다양한 컴포넌트에 본 논문의 기법과 도구를 적용시켜 보는 것과 프로덕트 라인 아키텍처 및 커넥터 패턴을 이용하여 응용 프로그램 전체의 자동 생성 기법에 대한 연구를 수행하는 것이다.

참고문헌

[1] P. Clements and Linda Northrop, "Software Product Lines: Practice and Patterns", Addison Wesley, 2002.

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

[3] J. C. Cleaveland, "Program Generators with XML and Java", Prentice Hall, 2001.

[4] C. Kang, S. Kim, J. Lee, K. Kim, E. Shin, M. Huh, "FORM: A Feature-Oriented Reuse Mehtod with Domain Specific Reference Architectures", Annals of Software Engineering, 5: pp.143~168, 1998.

[5] 송재승, 김민성, 박수용, " 프로덕트 라인 개발에서 피쳐 모델의 명세화 기법" , 한국정보과학회논문지: 소프트웨어 및 응용, 제30권, 11호, 2003년, pp.1001~1014.

[6] H. Zhang, S. Jarzab and S. M. Swe, "XVCL Approacn to Separating Concerns in Product Family Assets", Proceedings of International Conference on Generative Component-Based Software Engineering (GCSE 2001), 2001.

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