특성 모델과 XML을 이용한 컴포넌트 코드 생성기

권미옥[0] 최승훈
덕성여자대학교 전산학과
(miok[0], csh)@duksung.ac.kr

Component Code Generator Using the Feature Model and XML

Miok Kwon[0] Seung-Hoon Choi
Dept. of Computer Science, Duksung Women's University

요 약

최근 소프트웨어 위기를 극복하기 위한 객체 지향 개발 방법론의 한계성이 나타남에 따라, 컴포넌트 기반의 소프트웨어 공학에 대한 연구가 활발히 진행되고 있다. 효율적인 컴포넌트 재사용을 위해서는, 컴포넌트 개발 시 하나의 컴포넌트 개발에 집중하기 보다는 공통된 특징을 공유하는 컴포넌트 패밀리 개발에 초점을 맞추어야 한다. 본 논문에서는 컴포넌트 패밀리 구축을 위한 도메인 공학과 특성 모델(Feature Model)을 기반으로, XML 명세서를 받아들여 사용자가 원하는 컴포넌트를 자동 생성하는 컴포넌트 코드 생성기와 재사용 프로세스를 제안한다. 컴포넌트 재사용자가 특정 컴포넌트 패밀리의 특성 모델에서 필요한 특성을 선택하면 컴포넌트 코드 생성기는 재사용자의 목적과 환경에 적합한 코드를 자동 생성한다.

1. 서론 및 연구 배경

최근 소프트웨어 위기(Software Crisis)를 극복하기 위한 객체지향 개발 방법론의 한계성이 나타남에 따라, 컴포넌트 기반의 소프트웨어 공학에 대한 연구가 활발히 진행되고 있다. 이는 컴포넌트 재사용성과 독립성을 보장함으로써 소프트웨어의 개발 생산성 향상에 기여하고 있다.

효율적인 컴포넌트 재사용을 위해서는, 컴포넌트 개발자가 구성 용이성(configurability)을 지원하는 컴포넌트를 개발함으로써, 컴포넌트 재사용자가 컴포넌트 재사용 시 자신의 목적과 환경에 적합하도록 변경할 수 있어야 한다. 이를 위해서는, 하나의 컴포넌트 개발에 집중하기 보다는 공통된 특징을 공유하는 컴포넌트 패밀리 개발에 초점을 맞추어야 한다. 이러한 컴포넌트 패밀리 구축 프로세스는 도메인 공학(Domain Engineerign)과 특성 모델(Feature Model)에 기반을 둔다.

시스템 패밀리 또는 컴포넌트 패밀리를 생성하기 위한 여러 가지 도메인 공학 기법들이 제안되었다[1][2]. 특성 중심 도메인 분석(FODA)[1] 기법에서는 특정 도메인 내에 존재하는 컴포넌트 패밀리의 공통점과 차이점을 특성 모델을 통하여 식별, 조직한다. 각 특성(feature)은 구체적인 컴포넌트의 기능과 형태에 직접적인 영향을 미칠 수 있는 중요한 특징들을 나타내는 것으로 필수적(mandatory), 선택적(optional), 택일적(alternative) 속성으로 분류된다. 특정 컴포넌트 패밀리의 특성 모델의 한 인스턴스는 구체적인 컴포넌트의 기능(capability)을 서술한다.

특성 모델을 컴포넌트 생성기 구현에 적용하는 경우 다음과 같은 이점이 있다.

1) 컴포넌트 생성 시 필요한 객체의 추출이 쉬워진다.
2) 영역 중심의 재사용성이 높은 객체를 추출할 수 있다.
3) 컴포넌트의 유지보수성, 이해도, 재사용성이 향상된다.

한편, 네트워크나 웹 상에서의 데이터 교환을 목적으로 개발된 XML(eXtensible Markup Language)은 도메인 지식의 표현에 대한 표준 및 자기 서술적인 기능을 제공함으로써 데이터검색의 효율성과 보다 단순한 응용프로그램 개발을 가능하도록 한다. 이러한 XML 관련 기술의 적용 분야가 점점 확대되고 있다. 특히, 소프트웨어 개발의 생산성을 높이기 위하여 최근 활발히 연구되는 소프트웨어 자동 생성 프로그래밍(Generative Programming) 분야에서의 적용 가능성도 보여지고 있다[3][4].

[3]에서는, 도메인 공학을 통해 시스템 패밀리의 공통점(commonality)과 차이점(variability)을 파악한 후, 가변점 (variation point)을 XML 명세서로 표현하여 구체적인 응용 프로그램을 자동으로 생성할 수 있는 프로그램 생성기를 제안하였다. 이러한 기술들은 컴포넌트 패밀리를 파악하고 모델링 하여 컴포넌트 생성 시스템을 구축하는 데에도 그대로 적용될 수 있다.

본 논문에서는 특성모델과 XML을 이용한 컴포넌트 코드 생성기 구축 방법과 재사용 프로세스를 제안한다. 이러한 컴포넌트 코드 생성기는 컴포넌트 재사용시에 컴포넌트 자동 생성을 통한 구성 용이성을 지원함으로써 재사용자가 자신의 목적과 환경에 보다 적합한 컴포넌트를 생성할 수 있도록 해 준다.

본 논문의 전체적인 구성은 다음과 같다. 제 2 장에서 연결 리스트(Linked List) 컴포넌트를 위한 코드 생성기를 사례 연구로 하여 컴포넌트 코드 생성기 구축 과정을 기술하고, 제 3 장에서 컴포넌트 코드 생성기를 이용한 재사용 과정을 보인다. 마지막으로, 제 4 장에서 결론 및 향후 연구과제를 기술한다.

2. 컴포넌트 코드 생성기 구축

본 장에서는 [5]에서 parametric polymorphism과 C++의 템플릿 클래스를 이용하여 컴포넌트 자동 생성을 구현했던 연결 리스트를 사례 연구로 하여, 특성 모델과 XML에 기반한 컴포넌트 코드 생성기의 구축 과정을 기술한다.

2.1 특성 모델 (Feature Model)

컴포넌트 코드 생성기를 구축하기 위해서는, 먼저 도메인 공학 기법을 이용하여 컴포넌트 패밀리를 식별한 후, 하나의 컴포넌트 패밀리 안에 존재하는 컴포넌트들이 가지는 공통점과 차이점들을 분석한다. 정확한 결정 영역(decision space)을 기술하기 위해

컴포넌트 패밀리가 가지는 공통점은 필수적(mandatory) 특성으로, 차이점은 선택적(optional) 또는 택일적(alternative)인 특성으로 분류하고, 이를 바탕으로 특성 모델을 작성한다. 연결 리스트에 대한 특성 모델은 그림1과 같다.

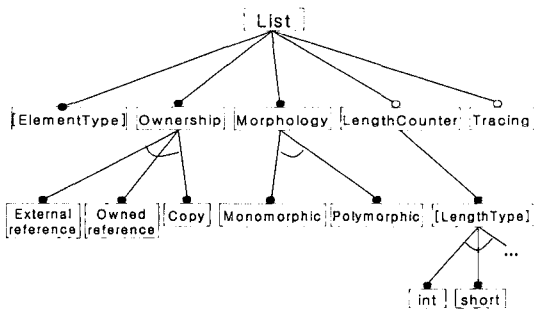

그림 1. 연결 리스트의 특성 모델.

그림1에서 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, LengthCounter는 연결 리스트의 길이를 계산하는 연산의 포함 여부를 의미하고, Tracing 특성은 리스트의 동작의 추적 기능 여부를 나타낸다. 이 특성 모델에 의하면, 자동 생성될 수 있는 구체적인 연결 리스트들은 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.

2.2 XML 형태의 구성(Configuration) 파일

2.2.1 컴포넌트 구성 규칙 명세서

특성 모델로부터 재사용자가 선택한 특성에 따라서 어떠한 구성 요소들이 구체적인 컴포넌트 생성 시 포함되어야 할 것인가에 대한 규칙을 정의해놓은 XML 파일이다. 즉, 컴포넌트 구성 규칙 명세서는 구체적인 컴포넌트 생성 시 컴포넌트의 구성을 결정하는 규칙을 포함한다. 컴포넌트 구성 규칙 명세서를 위한 DTD는 그림 2와 같다.

```
<!ELEMENT Config (category*)>

<!ELEMENT category (ImplCom*)>

<!ELEMENT ImplCom (OR_cond*)>

<!ELEMENT OR_cond (AND_cond*)>

<!ELEMENT AND_cond (#PCDATA)>

<!ATTLIST category name CDATA #REQUIRED>
<!ATTLIST ImplCom name CDATA #REQUIRED>
```

그림 2. Configuration_Rule.dtd

2.2.2 컴포넌트 구성 명세서

컴포넌트 구성 명세서는 하나의 컴포넌트 패밀리로부터 구체적인 컴포넌트를 생성하기위한 변이점을 표현한 XML 파일이다. 이는 특성 모델에서 재사용자에 의해서 선택되어진 특성들로 구성된다. 컴포넌트 구성 명세서를 위한 DTD는 그림 3과 같다.

```
<!ELEMENT Tree (Root, Node*)>

<!ELEMENT Root (#PCDATA)>

<!ELEMENT Node (name, type, parent, value?)>

<!ELEMENT name (#PCDATA)>
<!ELEMENT type (#PCDATA)>
<!ELEMENT parent (#PCDATA)>
<!ELEMENT value (#PCDATA)>
```

그림 3. Com_Config.dtd

2.3 컴포넌트 코드 생성기 구현

컴포넌트 코드 생성기는 그림 4와 같이 크게 다음 3가지 과정을 통하여 컴포넌트 코드를 자동 생성한다.

1) 특성모델에서 재사용자에 의해 선택되어진 특성들을 표현하는 컴포넌트 구성 명세서를 읽어들인다.
2) 컴포넌트 구성 명세서와 컴포넌트 구성 규칙 명세서를 바탕으로 컴포넌트 저장소에서 구체적인 컴포넌트 생성에 필요한 재사용 가능한 부품들을 선택한다.
3) 컴포넌트 구성 명세서에 명시된 특성 값과 선택된 부품들을 바탕으로 구체적인 컴포넌트 코드를 자동 생성한다.

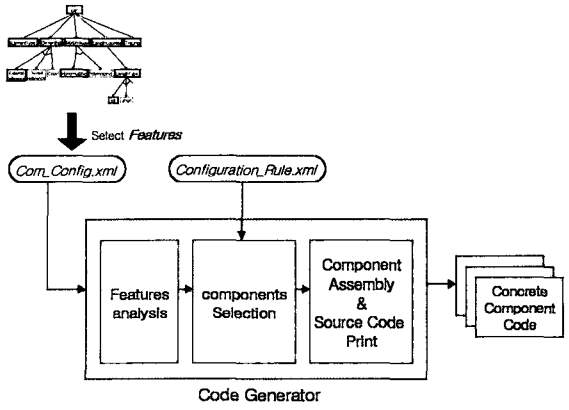

그림 4. 컴포넌트 코드 생성기의 구조.

3. 컴포넌트 코드 생성기를 이용한 재사용

본 장에서는 연결 리스트 코드 생성기를 예로 들어, 컴포넌트 코드 생성기를 이용한 재사용 과정을 설명한다. 컴포넌트 코드 생성기를 이용한 컴포넌트 코드 자동 생성 과정은 다음과 같다.

1) 재사용자가 자동 생성하기를 원하는 컴포넌트가 속한 컴포넌트 패밀리를 결정한다. (예 : Linked List)
2) 컴포넌트 개발자에 의해 미리 정의된 특성 모델로부터, 재사용자는 구체적인 컴포넌트 생성에 필요한 필수적, 선택적, 택일적인 특성을 선택하고 필요한 커스터마이징 값을 입력하여 컴포넌트 구성 명세서 XML 파일을 자동 생성한다. 그림 5는 'Base' 형만을 리스트의 원소로 갖고, LengthCouter를 가지며, Trace가 가능한 리스트에 대한 구성 명세서의 일부분을 보여준다.

```
<!DOCTYPE Tree SYSTEM "file:Configuration.dtd">
<Tree>
   <Root>List</Root>
   <Node>
      <name>ElementType</name>
      <type>Mandatory</type>
      <parent>List</parent>
      <value>Base</value>
   </Node>
      ..........
   <Node>
      <name>External reference</name>
      <type>Alternative</type>
      <parent>Ownership</parent>
   </Node>
      ..........
</Tree>
```

그림 5. List_Config.xml

3) 컴포넌트 코드 생성기는 컴포넌트 구성 명세서를 입력 받아, XML형태의 구성 규칙 명세서를 바탕으로 컴포넌트 저장소에서 필요한 부품들을 선택한다

```
<!DOCTYPE Config SYSTEM "file:Configuration_Rule.dtd">
<Config>
   <category name="Destroyer">
     <ImplCom name="ElementDestroyer">
       <OR_cond>
         <AND_cond>Copy</AND_cond>
       </OR_cond>
       <OR_cond>
         <AND_cond>Owned reference</AND_cond>
       </OR_cond>
     </ImplCom>
     <ImplCom name="EmptyDestroyer">
       ..........
     </ImplCom>
   </category>
   ..........
</Config>
```

그림 6. Configuration_Rule.xml

그림 6은, 재사용자가 선택한 특성에 따라 연결 리스트의 구성 요소인 Destroyer, Copier, TypeChecker들 중 어떤 것들이 구체적인 연결 리스트 생성 시 포함될 것인가를 나타낸다. 코드 생성기는 이 파일에 표현되어 있는 구성 규칙을 이용해서 선택된 특성에 따라 구체적인 연결 리스트 코드 생성 시 포함해야 할 부품들을 결정한다.

4) 컴포넌트 코드 생성기는 컴포넌트 구성 명세서와 선택된 부품들을 이용하여, 구체적인 컴포넌트 코드를 자동 생성한다. 그림 7은, 그림5의 구성 명세서에 따라 생성한 연결 리스트 코드이다.

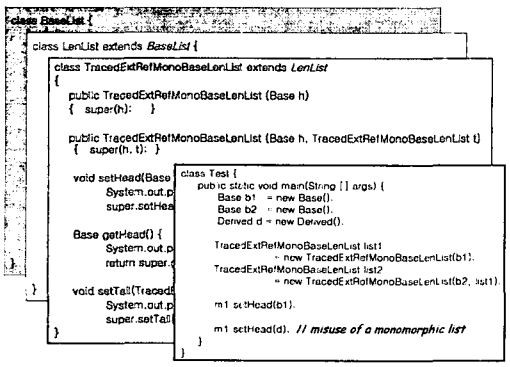

그림 7. 자동 생성된 연결 리스트 코드

그림7을 보면, 연결 리스트 코드 생성기는 리스트 패밀리의 공통점만을 기능으로 가지는 BaseList를 상속받아, Length Counter, Trace의 기능을 추가시킨 TracedExtRefMonoBaseLenList의 코드를 자동 생성함을 알 수 있다. 생성된 리스트는 리스트 구성 명세서에 명시된 커스터마이징 값과 선택되어진 부품을 사용한다. Test 클래스의 main( ) 메소드는 생성된 연결 리스트를 시험하기 위한 코드의 일부분을 보여준다.

컴포넌트 생성기를 이용하면, 컴포넌트 재사용자는 특정 컴포넌트 패밀리의 특성 모델에서 필요한 특성을 선택하고, 특성의 커스터마이징 데이터만 입력하면 자신의 목적과 환경에 적합한 코드를 자동 생성함을 알 수 있다.

특성 모델과 XML을 이용한 컴포넌트 코드 생성기의 장점은 다음과 같다.

1) 보다 추상적인 모델인 특성 모델을 통해서 컴포넌트를 바라볼 수 있으므로, 재사용자는 컴포넌트 구현에 대한 지식이 필요없이 컴포넌트의 전체적인 의미(semantic)를 쉽게 이해할 수 있다.

2) 특성 모델이 XML 형태로 표현되기 때문에, 컴포넌트 정보에 대한 표준화를 제공하여 효율적인 컴포넌트 정보 저장과 검색을 가능하게 한다.

3) 구성 규칙이 XML 형태로 표현되기 때문에, 컴포넌트 개발자는 구체적인 컴포넌트 생성 시 포함되어야 할 부품들을 변경하기 위해서 이 구성 규칙 파일만을 수정하면 된다.

4) 소스 코드 생성 시, 여러 가지 다른 기능을 수행하는 코드를 삽입하거나 생성할 수 있다. (예: 시험 코드 자동 생성)

4. 결론

본 논문에서는 특성 모델과 XML을 기반으로 하는 컴포넌트 코드 생성기와 재사용 프로세스를 제안하였다. 컴포넌트 코드 생성기 개발자는, 컴포넌트 요구 사항이나 기능들을 바탕으로 하나의 컴포넌트 패밀리 안에 존재하는 컴포넌트들이 가지는 공통점과 차이점들을 분석하여 특성 모델을 정의한다. 그리고 재사용자가 선택한 특성에 따라 어떤 구성 요소들을 포함할 것인가를 나타내는 XML 형태의 컴포넌트의 구성 규칙 명세서를 작성한다. 컴포넌트 재사용자는 재사용시 특성 모델로부터 자신의 목적과 환경에 맞는 특성들을 선택하면 그에 대한 코드가 자동 생성된다.

본 논문의 컴포넌트 코드 생성기와 재사용 프로세스는, 컴포넌트 자동 생성을 통한 구성 용이성을 지원함으로써 컴포넌트 재사용 시 생산성을 높여주며, XML이 가지는 다양한 장점들 – 도메인 지식의 표준 형식, 자기 기술적인 특징 등 – 을 제공한다.

향후 과제로서, 테스트 코드 자동 생성, 컴포넌트 생성기 개발 지원 도구 등에 대한 연구가 필요하다.

참고 문헌

[1] K. Kang, S. Cohen, J.Hess, W.Nowak and S. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study", Technical Report. CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, November, 1990.

[2] M. Simos, D. Creps, C. Klinger, L. Levine, and D. Allemang, "Organization Domain Modeling(ODM) Guidebook", Version 2.0, Informal Technical Report for STARS, STRAS-VC-A025/011/00, June 14, 1996, available from http://domain-modling.com .

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

[4] Miok kwon and Seung-Hoon Choi, "Component Generating System based on XML", Korean Society For Internet Information, 2001.

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