온톨로지 기반의 특성 모델 검증 도구

김민경, 송은총, 한지희, 최승훈
덕성여자대학교 컴퓨터시스템과
e-mail:jiheez@duksung.ac.kr

Feature Model Validation Tool based on Ontology

Min-Kyung Kim, Eun Chong Song, Ji Hee Han, Seung-Hoon Choi
*Dept of Computer System, Duksung Women's University

요약

소프트웨어 제품 라인 개발 패러다임은 관련 제품들 사이의 공통점과 차이점을 이용해 보다 전략적인 재사용을 가능하게 함으로써 소프트웨어 개발 생산성을 높여 주는 개발 방법론이다. 공통점과 차이점을 분석하고 모델링하기 위해 가장 중요한 모델이 특성 모델이다. 특성 모델은 규모가 커짐에 따라 오류를 포함할 가능성이 커지며 이를 검증하기 위한 자동화된 도구가 필요하다. 본 논문에서는 온톨로지를 자바 언어로 구현 가능하게 해주는 Protege API, OWL기반의 시맨틱 웹 규칙 언어인 SWRL, 규칙 추론 엔진인 Pellet Reasoner 등의 기술을 이용한 특성 모델 검증 도구를 제안한다.

1. 서론 및 연구배경

소프트웨어 제품 라인 개발 패러다임은 소프트웨어 개발 단계 초기에 소프트웨어 제품군에 속하는 멤버들 사이의 공통점과 차이점을 미리 예측하고 분석함으로 보다 전략적인 재사용이 가능하도록 해 소프트웨어 개발 생산성을 향상시키는 방법론으로 각광을 받고 있다[1]. 소프트웨어 제품 라인 개발 시 공통점과 차이점을 분석하고 모델링하기 위해 가장 중요한 것이 특성 모델[2]이다. 특성 모델의 규모가 커지고 제한조건이 많아지면 오류를 포함할 가능성이 커진다. 이러한 오류들을 수작업으로 찾아내기란 불가능하며 성공적인 소프트웨어 제품 라인 개발을 위해서는 특성 모델을 검증하기 위한 자동화 도구가 필수적이다.

본 논문에서는 [3]에서 소프트웨어 제품 라인의 표준 문제로서 제안한 Graph Product Line 특성 모델을 예제로 하여 온톨로지 기반의 특성 모델 검증 도구를 제안한다. 본 도구는 특성 모델을 Protege API[4]를 이용해 OWL(Web Ontology Language)언어[5]로 표현한다. 특성 모델의 오류를 발견하기 위한 규칙은 시맨틱 웹 규칙 언어인 SWRL(Semantic Web Rule Language)[6]로 정의하고, Pellet 규칙 추론 엔진[7]을 이용하여 특성 모델을 검증한다.

본 논문의 구성은 다음과 같다. 제 2 장에서 특성 모델에서의 오류를 설명하고, 제 3 장에서 온톨로지를 이용한 특성 모델과 검증 규칙에 대해 기술한다. 제 4 장에서 GPL 예제를 이용하여 특성 모델 검증 도구를 살펴 본 후, 제 5 장에서 결론 및 향후 연구 과제를 기술한다.

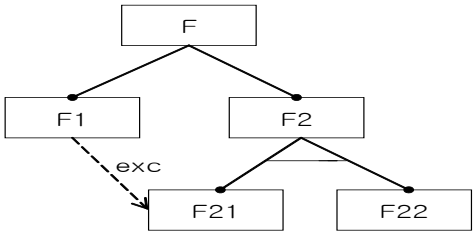
2. 특성 모델 오류

[8]에서는 특성 모델이 포함할 수 있는 오류로서 dead feature와 full-mandatory feature를 정의하였다. 각 오류의 의미는 다음과 같다.

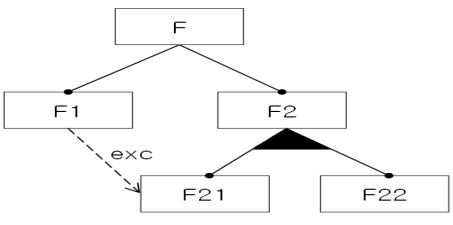
-dead feature : 어떠한 특성구성에도 포함될 수 없는 특성

-full mandatory feature : mandatory특성이 아니면서 부모가 특성 구성에 선택된 경우에는 반드시 특성구성에 포함되어야 하는 특성

그림 1과 2는 [8]에서 제안한 오류 중 일부를 보여준다.



(그림1) Dead Feature 오류

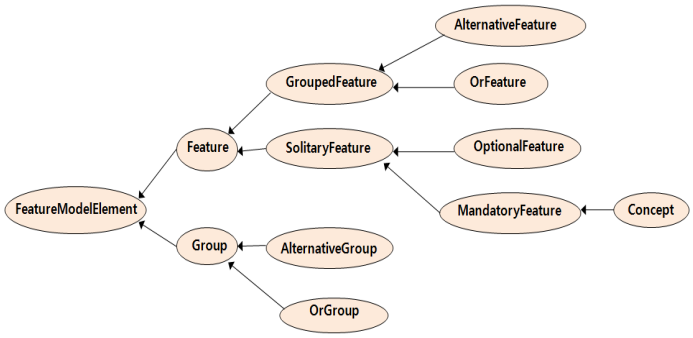


(그림2) Full Mandatory Feature 오류

3. 특성 모델 온톨로지

3.1 클래스 및 속성

특성 모델을 OWL기반의 온톨로지[9]로 정의하기 위한 클래스 계층 구조는 다음과 같다. 특성을 나타내는 Feature 클래스는, 독자적으로 존재하는 특성을 나타내는 SolitaryFeature 클래스와 그룹의 멤버로 존재하는 특성을 나타내는 GroupedFeature 클래스 두 개로 분류된다. 특성 모델에 존재하는 그룹으로는 택일적 그룹(AlternativeGroup 클래스)과 OR 그룹(OrGroup 클래스)이 있다. 그림3에서 Feature 클래스와 Group 클래스는 FeatureModelElement 클래스를 상속한다.



(그림3) 특성 모델을 위한 클래스 계층 구조

또한 특성 모델의 트리 구조를 표현하기 위해 여러 가지 객체 속성(object Property)이 필요하다. 표1은 특성 모델 온톨로지에서의 객체 속성들과 그 의미 및 정의역(domain)과 치역(range)을 보여준다.

객체 속성	의미(Domain/Range)
hasParent	한 특성의 부모 특성을 의미함 (D: SolitaryFeature/ R: Feature)
hasChild	한 특성의 자식 특성을 의미함 (D: Feature/ R: SolitaryFeature)
hasGroup	한 특성이 가지는 그룹을 의미함 (D: Feature/ R: Group)
isGroupOf	한 그룹의 부모 특성을 의미함 (D: Group/ R: Feature)
hasMember	한 그룹의 멤버를 의미함 (D: Group/ R: GroupedFeature)
isMemberOf	한 멤버가 속하는 그룹을 의미함 (D: GroupedFeature/ R: Group)
requires	한 특성과 requires 관계에 있는 특성을 의미함 (D: Feature/ R: Feature)
excludes	한 특성과 excludes 관계에 있는 특성을 의미함 (D: Feature/ R: Feature)

(표1) 특성 모델 온톨로지에서의 객체 속성들

3.2 특성 모델 검증을 위한 SWRL 규칙

특성 모델에 있는 오류를 검사하기 위해서는 그림1, 그림2에 정의되어 있는 모든 오류들에 대하여 이를 검증하기 위한 규칙을 정의해야 한다. OWL규칙은 OWL클래스, 속성, 인스턴스, 데이터 값 등을 이용하여 작성하며 '→' 의 왼쪽부분의 조건이 참이 되면 오른쪽 부분의 인스턴스를 생성함으로 오류를 일으킨다는 사실을 추론한다. 그림4는 그림1에서의 특성 모델 오류를 검증하기 위한 규칙을 보여준다. 이 규칙은 필수적 특성 ?f2의 택일적 자식 ?f21을 또 다른 필수적 특성 ?f1이 excludes하는 경우 ?f21은 dead feature임을 나타낸다. 그림5는 그림2에서의 특성 모델 오류를 검증하기 위한 규칙을 보여준다. 필수적 특성 ?f2의 OR자식이 ?f21과 ?f22 두 개만 존재하고, 필수적 특성 ?f1이 OR 특성 ?f21을 excludes하는 경우 ?f22는 full mandatory feature임을 나타낸다.

```
MandatoryFeature(?f2) ∧ hasGroup(?f2, ?group) ∧
hasMember(?group, ?f21)∧AlternativeFeature(?f21)∧
MandatoryFeature(?f1) ∧ excludes(?f1, ?f21)
  → DeadFeature1(?f21)
```

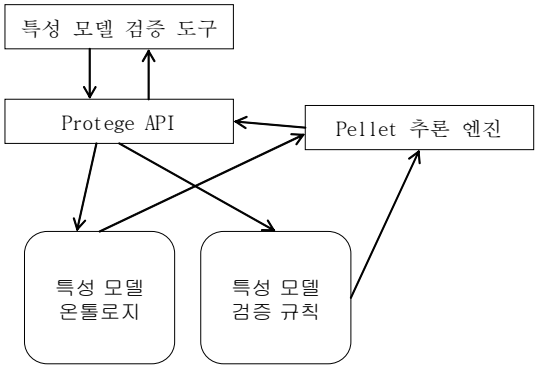
(그림4) 그림1 오류를 검증하기 위한 SWRL규칙

```
MandatoryFeature(?f2) ∧ hasGroup(?f2,?group) ∧
hasMember(?group, ?f21) ∧ OrFeature(?f21) ∧
hasSameGroupMemberCount(?f21, 1) ∧
MandatoryFeature(?f1) ∧ excludes(?f1, ?f21)  ∧
OrFeature(?f22) ∧ hasSameGroupMember(?f21, ?f22)
 → FullMandatoryFeature3(?f22)
```

(그림5) 그림2 오류를 검증하기

4. 특성 모델 검증 도구

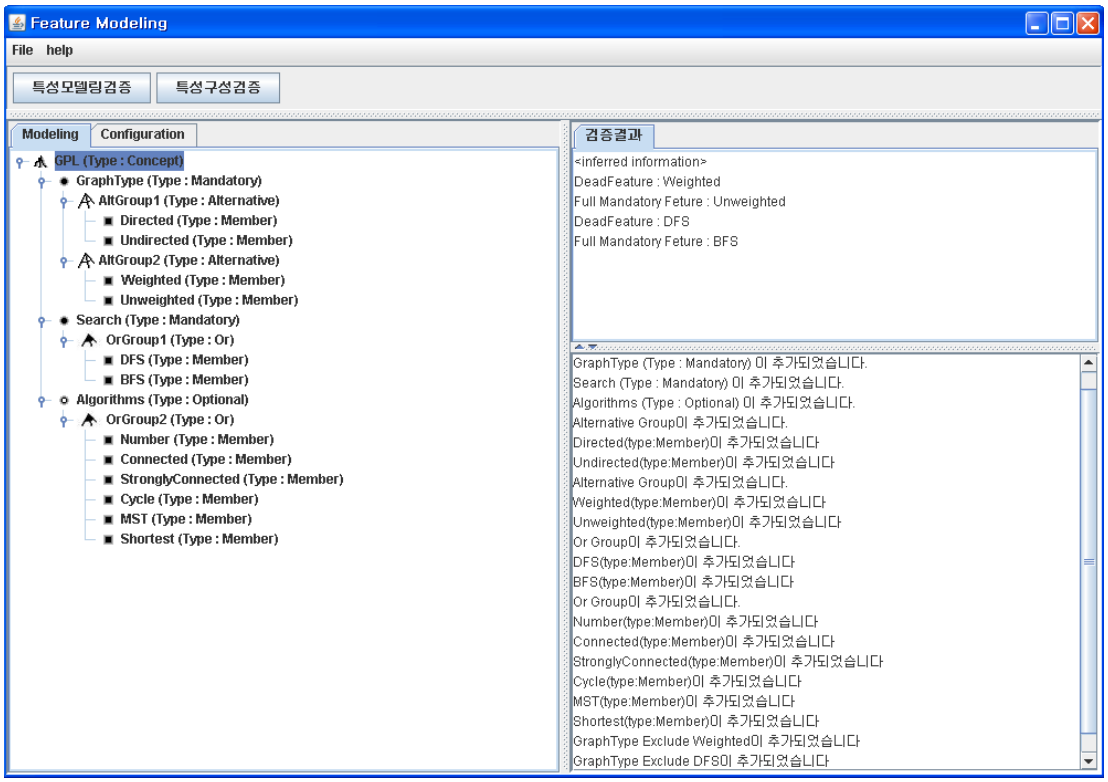
본 논문의 온톨로지 기반 특성 모델 도구의 전체적인 구조는 그림6과 같다. 특성 모델 도구는 먼저 Protege API를 이용하여 특성 모델 온톨로지 중에서 클래스와 속성들을 정의한다. 사용자는 특성 모델 도구를 이용하여 특성 모델을 작성하며 작성된 특성 모델은 Protege API를 이용하여 온톨로지 중 인스턴스로 변환된다. 이 후에 특성 모델 도구는 특성 모델 온톨로지, 특성 모델 검증 규칙, Pellet 추론 엔진을 이용하여 특성 모델을 검증하고 특성 모델에 포함된 오류들을 발견한다. 그림7은 본 논문에서 구현한 특성 모델 검증 도구의 사용자 인터페이스를 보여준다. 그림7에서 왼쪽 부분은 본 논문에서 사례 연구로 이용한 Graph Product Line(GPL) 특성 모델을 보여준다. 각 아이콘의 의미는 표2와 같다.



(그림6) 특성 모델 도구 인터페이스

특성 모델 검증 도구의 오류 검증 기능을 테스트하기 위해 기존의 GPL 특성 모델에 몇 가지 오류를 추가하였다. 택일적 특성이었던 "Search" 특성을 mandatory로 변경하고, "Search" 특성이 가지는 그룹을 "Or"로 변경하였다. 또한, "GraphType" 특성이 "Weighted", "DFS" 특성들과 exclude 관계를 가지게 하였다(그림1과 그림2에 해당하는 오류를 추가했음). 그림8은 이와 같은 특성들 사이의 관계를 설정하는 과정을 보여준다.

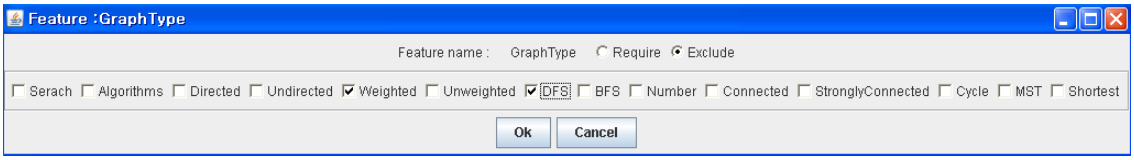
그림7의 오른쪽 부분은 Pellet Reasoner를 이용하여 GPL 특성 모델을 검증한 결과를 보여준다. GPL에 포함시켰던 오류들을 정확히 검증해 내었음을 확인할 수 있다.



(그림7) 특성 모델 검증 도구

아이콘	유형	의미
	Concept	모델의 가장 상위 Feature이며, 특성 모델의 이름을 나타낸다.
	Mandatory	Feature의 한 종류로서, 반드시 선택되어야 한다.
	Optional	Feature의 한 종류로서, 선택 여부를 선택할 수 있다.
	Or(group)	자식 노드로 존재하는 노드들 중 1개 이상을 선택 할 수 있다.
	Alternative(group)	자식 노드로 존재하는 노드들 중 1개만을 선택 할 수 있다.
	member	그룹의 하위 노드로 오는 Feature를 나타낸다.

(표2) 각 아이콘의 유형과 의미



(그림8) [그림7]특성 모델에 관계(오류) 설정

5. 결론 및 향후 연구

소프트웨어 제품 라인 패러다임은 소프트웨어들 사이의 공통점과 차이점을 이용한 효율적인 소프트웨어 개발 방법론이다. 특정 도메인에서 제품들 사이의 공통점과 차이점을 모델링 하는 것은 소프트웨어 제품 라인 공학에서 가장 중요한 단계이며 이 때 사용되는 핵심 모델이 특성 모델이다.

특성의 종류가 많아지고 특성들 사이에 존재하는 조건이 복잡해지면, 특성 모델에는 오류가 포함될 가능성이 커지며 이를 검증하기 위한 자동화된 기법이 필요하다. 본 논문에서는 특성 모델을 OWL 언어를 이용하여 온토로지로 표현하고, 오류 검증을 위한 규칙을 SWRL(Semantic Web Rule Language)을 이용하여 정의한 후, Pellet 규칙 추론 엔진을 이용하여 특성 모델을 검증하였다. 이 모든 작업을 위해 Protege API를 이용하였다.

본 논문의 특성 모델 검증 도구는 새로운 검증 규칙을 추가하기가 쉬우며 자바 언어로 구현되었기 때문에 다른 소프트웨어 개발 환경과 쉽게 연동 가능하다. 또한, 새로운 시맨틱 웹 기술(예를 들면 SQWRL)과도 쉽게 통합될 수 있다.

향후 연구 과제로서 특성 구성(feature configuration) 검증에 대한 연구, 특정 제품에 대한 자동 코드 생성, 시맨틱 웹 기술에 대한 모델링 지원 방법 등에 대한 연구가 있다.

참고문헌

[1] P. Clements and L.Northrop, "Software Product Lines: Practices and Patterns", Addison Wesly, 2002.

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

[3] Roberto E.Lopez-Herrejon and Don S. Batory. A standard problem for evaluating productline methodologies. In Proceedings of the Thired International Conference on Generative and Component-Based Software Engineering, pages 10-24, Erfult, Germany, September 2001.

[4] http://protege.stanford.edu/

[5] http://www.w3.org/TR/owl-features/

[6] http://www.w3.org/Submission/SWRL/

[7] http://protegewiki.stanford.edu/wiki/ProtegeReasonerAPI

[8] P. Trinidad, D. Benavides, A. Durán, A. Ruiz-Cortés and M. Toro, "Automated Error Analysis of Feature Models", Journal of Systems and Software(in press), 2008.

[9] Park, J, "Ontology" in Management Information Systems, G.B. Davis (ed.) Cambridge, Massachusetts, Blackwell Publishing, Vol, VII, 2005, pp.233-236.