규칙 기반 특성 모델 검증 도구☆

Rule-based Feature Model Validation Tool

최 승 훈*
Seung Hoon Choi

요 약

특성 모델(Feature Model)은 소프트웨어 제품 라인 개발 시 도메인 공학 단계에서 제품들 사이의 공통점과 차이점을 모델링하는데 널리 사용된다. 특성 모델의 오류 또는 불일치성에 대한 발견 및 수정은 성공적인 소프트웨어 제품 라인 공학을 위해서 필수적이다. 특성 모델의 검증을 효과적으로 수행하기 위해서는 자동화된 도구의 도움이 필요하다. 본 논문에서는 JESS 규칙 기반 시스템을 이용하여 특성 모델의 유효성을 검증하는 기법을 기술하고 이를 이용한 특성 모델 검증 도구를 제안한다. 본 논문의 도구는 특성 모델링 작업 시 실시간으로 특성 모델을 검증하여 오류의 존재 여부와 오류의 원인에 대한 설명을 제공함으로써 오류 없는 특성 모델을 생성할 수 있도록 해 준다. 특성 모델 검증 기법에 규칙 기반 시스템을 이용한 경우는 본 논문이 최초의 시도로 사료된다.

Abstract

The feature models are widely used to model the commonalities and variabilities among the products in the domain engineering phase of software product line developments. The findings and corrections of the errors or consistencies in the feature models are essential to the successful software product line engineering. The aids of the automated tools are needed to perform the validation of the feature models effectively. This paper describes the approach based on JESS rule-base system to validate the feature models and proposes the feature model validation tool using this approach. The tool of this paper validates the feature models in real-time when modeling the feature models. Then it provides the information on existence of errors and the explanations on causes of those errors, which allows the feature modeler to create the error-free feature models. This attempt to validate the feature model using the rule-based system is supposed to be the first time in this research field.

☞ Keyword : software product lines, feature model validation, rule-based system, 소프트웨어 제품 라인, 특성 모델 검증, 규칙 기반 시스템

1. 서론 및 연구배경

소프트웨어 제품 라인(Software Product Lines)이란, 공통된 핵심 소프트웨어 자산(core software asset)들로부터 특정 목표나 시장을 위해서 개발된 소프트웨어 집약 시스템들의 집합을 의미한다[1]. 소프트웨어 제품 라인의 목적은 소프트웨어 개발 단계 초기에 소프트웨어 패밀리에 속하는 멤버들 사이의 차이점과 공통점을 미리 예측하고 분석함으로써 보다 전략적인 재사용이 가능하도록 하여 소프트웨어 개발 생산성을 향상시키고자 하는 것이다.

특성 모델(Feature Model)[2]은 소프트웨어 제품 라인 개발 시 도메인 공학 단계에서 가장 널리 사용되는 모델로서, 특정 도메인에서 제품들 사이의 공통된 개념들과 서로 다른 개념들을 모델링하는데 사용된다.

어플리케이션 공학(Application Engineering) 단계에서는 도메인 공학 단계에서 구축된 재사용 가능한 자산을 이용하여 구체적인 제품을 생산한다. 구체적인 제품 생산 시 먼저 특성 모델에 표현되어 있는 가변적 특성들 중에서 어떤 특성을 목표로 하는 제품에 포함시킬 것인가에 대한 요구 사항을 결정해야 한다. 특성 모델에 표현되어 있는 가변적 특성에 대해 특정 제품

* 종신회원 : 덕성여자대학교 컴퓨공학부 교수
csh@duksung.ac.kr
[2008/11/17 투고 - 2008/11/21 심사 - 2009/02/03 심사완료]
☆ 본 연구는 덕성여자대학교 2008년도 교내연구비 지원에 의해 수행되었음

에 포함될 특성들을 선택한 결과를 특성 구성(Feature Configuration)이라고 한다.

가변적 특성의 종류가 많아지고 제한 조건이 복잡해지면 특성 모델과 특성 구성이 오류를 포함할 가능성이 증가하며 이러한 오류를 수작업으로 검증하기는 거의 불가능하다. 따라서 성공적인 소프트웨어 제품 라인을 개발하기 위해서는 특성 모델과 특성 구성을 검증하기 위한 자동화 도구가 필수적이다.

[3]에 따르면 특성 모델의 자동화된 분석에 대한 연구가 어느 정도 이루어졌지만 특성 모델의 불일치성 발견, 불일치성에 대한 설명 등에 대한 연구가 더 필요하다고 주장한다.

본 논문에서는 JESS 규칙 기반 시스템을 이용하여 특성 모델에 존재하는 오류를 검증하기 위한 기법을 제안하고 이를 활용한 특성 모델 검증 도구를 구현한다.

본 논문의 구성은 다음과 같다. 제 2 장에서 특성 모델 검증에 대한 기존 관련 연구와 JESS 시스템에 대해 기술한다. 제 3 장에서 JESS 시스템을 이용한 특성 모델 검증 기법을 설명하고 제 4 장에서 구현된 특성 모델 검증 도구를 살펴본 후 본 논문의 기법을 평가하고 제 5 장에서 결론 및 향후 연구 과제를 기술한다.

2. 관련 연구

2.1 특성 모델 검증 기법

특성 모델의 자동화된 분석에 대한 연구가 최근 활발하게 진행되었다[4]. 특성 모델을 자동으로 분석하기 위한 오퍼레이션으로서, 특성 모델의 유효성(validation) 검증, 특성 모델로부터 생성 가능한 제품의 종류 및 개수 식별, 가변성 정도 및 공통성 정도 계산, dead feature의 발견, 특성 모델의 오류 설명 등이 있다.

특성 모델의 자동화된 분석을 위한 기법으로 명제논리학(propositional logic)에 기반을 둔 분석 방법[5] [6] [7], 서술 논리(description logic)에 기반을 둔 분석 기법[8] [9], 제한조건 프로그래밍(constraint programming)에 기반을 둔 분석 방법[10] [11] 등이 제안되었다. 이러한 분석 방법이 여러 가지 다양한 특성 모델 분석 방법을 제공하기는 하지만 주로 특성 구성과 관련된 분석 결과를 제공한다.

예를 들어 [7]에서는 특성 모델과 제한 조건(excludes, requires 등)을 grammar로 표현한 후 명제식(propositional formula)로 나타낸다. 그리고 나서 LTMS(Logic-Truth Maintenance Systems)를 이용하여 특성 구성 시 잘못된 제품 명세서가 작성되지 않도록 도와준다. 이 기법은 특성 구성에 대한 오류를 잘 검증해주는 반면, 특성 모델에 대해서는 단지 유효한지(valid) 아닌 지와 충돌을 일으키는 관계들만을 알려준다.

또한, [9]에서는 시맨틱 웹 기반의 특성 구성 검증 기법이 제안되었다. 이 기법은 특성 모델과 특성 구성을 Protégé 온톨로지 편집 도구를 이용하여 OWL 언어로 표현하고 RACER나 FaCT와 같은 추론 엔진을 사용하여 특성 구성이 일관적인지를 검증한다. 이 기법은 추론 엔진을 사용한다는 점에서 본 논문의 기법과 유사하지만 특성 모델에 대한 검증은 하지 않으며 특성 구성에 포함된 오류의 원인에 대해 자세한 설명을 제공하지 않는 단점이 있다.

현재 특성 구성에 대한 분석 기법이 다양하게 제안된 반면 특성 모델 자체의 오류를 분석하는 연구는 상대적으로 부족하다.

[12]에서는 특성 모델을 이진 결정 다이어그램(binary decision diagrams)으로 표현하고 이를 이용하여 특성 모델의 유효성 및 dead feature의 유무를 발견하는 기법을 제안하였지만 오류에 대한 설명은 제공하지 못한다.

[13]에서는 특성 모델을 first-order 논리 기반의 구조적 모델링 언어인 Alloy로 변환하고 SAT solver를 이용하여 무효한(void) 특성 모델의 원인이 되는 relationship을 분석하는 기법을 제안하였다.

[14]에서는 특성 모델 오류인 full-mandatory 특성과 dead feature를 정의하고, 이들을 발견하고 그 이유를 설명할 수 있는 기법을 '진단이론(Theory of Diagnosis)'와 'CSP(Constraint Satisfaction

Problem) 프로그래밍'을 이용하여 해결하는 방법을 제안하였다. 이 기법은 특성 모델의 일관성(consistency)를 검증하기 위하여 특성 모델을 CSP(Constraint Satisfaction Problem)로 변환하는 과정이 필요하다.

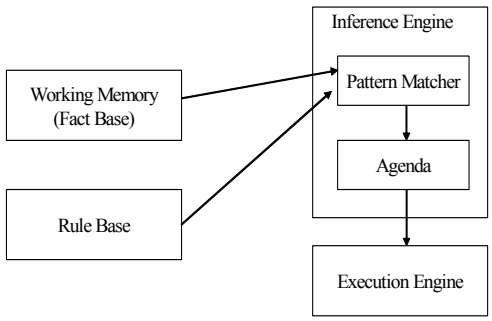
본 논문의 기법은 JESS 라는 자바 기반의 규칙 기반 시스템을 활용하기 때문에, 특성 모델러가 생성하는 오류를 실시간으로 검증하고 그 원인에 대한 설명을 제공한다. 또한, 새로운 오류가 정의되면 이에 대한 규칙만을 추가하면 되기 때문에 확장성이 뛰어나다. 또한 JESS 시스템이 자바 기반으로 구축되어 있기 때문에 기존의 자바 기반 소프트웨어 개발 툴과의 결합성이 좋다.

2.2 JESS 규칙 기반 시스템

JESS[15]은 Java Expert System Shell의 약자로써 Sandia 국립 연구소의 Ernest Friedman-Hill에 의해 자바 언어로 개발된 규칙 엔진이자 지식 기반 시스템 개발 환경이다. JESS 시스템을 사용하면 선언적인 규칙 형태로 제공되는 지식을 이용하여 새로운 사실을 추론하는 능력을 갖는 프로그램 개발이 가능하다. 또한, 자바 언어와 쉽게 결합 가능하여 JESS 언어에서 Java의 모든 API를 호출할 수 있을 뿐만 아니라 자바 언어에서 JESS 언어를 이용하여 논리적 프로그램을 작성할 수도 있다.

JESS 시스템의 구조는 그림1과 같다. 규칙이 적용될 대상이 되는 데이터는 fact라고 하는 기본 원소로 구성되며 working memory 또는 fact base에 존재한다. 본 논문에서의 특성 모델 결과에 대한 정보는 이 working memory에 존재한다.

Rule base에는 지식을 나타내는 규칙의 집합이 존재하며, if-then 형태로 정의된다. 이러한 규칙은 working memory에 존재하는 fact들에 대해서 if 절이 만족하였을 때 then에 정의된 행동을 실행한다. 본 논문에서는 특성 모델이 만족해야 할 제한 조건이 rule로 표현된다.



(그림 1) JESS 시스템 전체 구조

3. 특성 모델 검증

3.1 특성 모델에서의 오류

[16]에서는 특성 모델에 존재할 수 있는 결점(deficiencies)들을 세 가지 레벨로 분류하였다. 정보가 중복되어 모델링 되는 경우를 나타내는 redundancy, 의미 없는 정보가 모델링 되는 경우를 나타내는 anomalies, 어떠한 특성 구성도 생성할 수 없는 모순이 존재하는 경우를 나타내는 inconsistency가 그것이다.

[14]에서는 특성 모델이 포함할 수 있는 오류로서 inconsistency 레벨에 속하는 dead feature와 anomalies 레벨에 속하는 full-mandatory feature를 정의하였다. 각 오류의 의미는 다음과 같다.

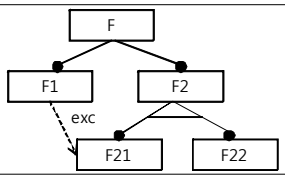
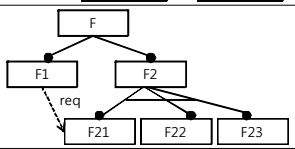
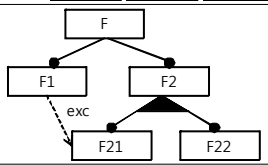
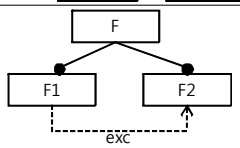
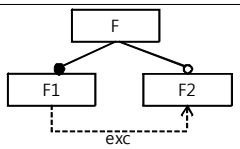
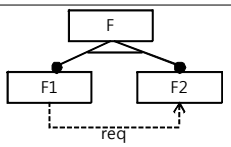
- dead feature: 어떠한 특성 구성에도 포함될 수 없는 특성
- full-mandatory feature: mandatory 특성이 아니면서 부모가 특성 구성에 선택된 경우에는 반드시 특성 구성에 포함되어야 하는 특성

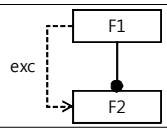
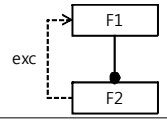
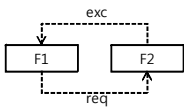
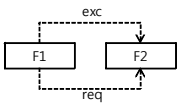
본 논문에서는 이러한 dead feature와 full-mandatory feature 오류들을 JESS 시스템을 이용하여 발견하고 설명하는 기법을 제안한다. 또한, redundancy 오류의 한 경우를 제시하고 본 논문에서 제안하는 시스템이 이 오류를 검증할 수 있도록 쉽게 확장됨을 보일 것이다.

표1은 dead feature 오류가 존재하는 경우를 보여주며, 표2은 full mandatory feature 오류가 발생하는 경우를 보여준다. 예를 들어, DF1 오류는 mandatory feature인 F1이 alternative feature의 하나인 F21을 excludes하는 제한조건을 가지기 때문에 F21은 어떠한 특성 구성에도 포함될 수 없으며 따라서 F21은 dead feature가 된다. 또한, FMF1은, F21이 dead feature가 되므로 같은 alternative feature group에 속하는 나머지 한 특성인 F22는 모든 특성 구성에 포함되어야 하므로 full mandatory feature가 된다.

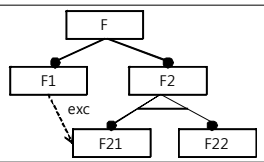
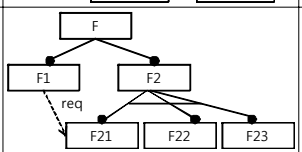
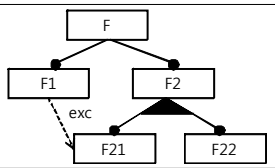
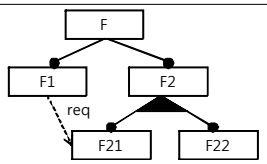
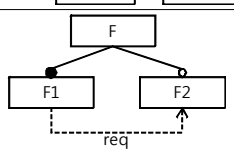
본 논문에서는 모든 경우의 오류에 대해서 이를 검증하기 위한 JESS 규칙을 정의한다.

(표 1) Dead Feature 오류들

ID	특성 모델	Dead Feature
DF1		F21
DF2		F22, F23
DF3		F21
DF4		F2
DF5		F2
DF6		F1, F2

DF7		F2
		F1
DF8		F1, F2
		F1, F2

(표 2) Full Mandatory Feature 오류들

ID	특성 모델	Full Mandatory Feature
FMF1		F22
FMF2		F21
FMF3		F22
FMF4		F21
FMF5		F2

3.2 특성 모델을 위한 JESS 템플릿

먼저 특성 모델을 표현하는데 사용될 fact들의 구조를 정의해야 한다. JESS에서는 (deftemplate)이라는 명령어를 이용하여 fact의 구조를 정의한다. 그림2는 특성 모델에 존재하는 각 특성의 정보를 표현하기 위한 템플릿 "feature"를 정의하는 JESS 코드 이다. feature 템플릿이 가지고 있는 슬롯의 의미는 표 3과 같다.

```
;;Meta model for Feature Model
(deftemplate feature ;; 특성 모델에서의 특성
   (slot name) ;; the name of the feature
   (slot type) ;; mandatory, optional, alternative, or
   (slot parent) ;; the name of parent feature
   (multislot sameGroupMembers)
   (multislot requires) ;; the names of required features
   (multislot excludes))    ;; the names of excluded features
```

(그림 2) 특성 모델을 위한 fact 구조 정의

(표 3) 'feature' 템플릿의 각 슬롯 의미

슬롯 이름	슬롯이 저장하는 정보
name	특성의 이름
type	특성의 타입으로서, mandatory, optional, alternative, or 중 하나의 값을 가진다.
parent	부모 특성의 이름
sameGroupMembers	특성의 type 슬롯의 값이 mandatory이거나 optional인 경우에는 nil 값을 가진다. 이 특성의 type 슬롯의 값이 alternative 또는 or인 경우에는, 같은 그룹에 속하는 특성들의 이름 리스트를 가진다.
requires	이 특성과 requires 관계의 있는 특성들의 이름 리스트
excludes	이 특성과 excludes 관계의 있는 특성들의 이름 리스트

본 논문에서 오류 검증을 위해 각 특성 모델은 JESS 템플릿에 따라 fact로 표현되고 working memory에 존재한다. 예를 들어 표1의 DF1의 특성 모델을 이 절에서 정의한 JESS 템플릿을 이용하여 저장한 경우의 facts의 모습은 그림3과 같다.

```
;; for feature F
(assert (feature (name F) (type mandatory) (parent nil) (sameGroupMembers nil ) (requires nil) (excludes nil)))
;; for feature F1
(assert (feature (name F1) (type mandatory) (parent F) (sameGroupMembers nil ) (requires nil) (excludes F21)))
;; for feature F2
(assert (feature (name F2) (type mandatory) (parent F) (sameGroupMembers nil ) (requires nil) (excludes nil)))
;; for feature F21 (Dead Feature)
(assert (feature (name F21) (type alternative) (parent F2) (sameGroupMembers F22 F23) (requires nil) (excludes f1)))
;; for feature F22
(assert (feature (name F22) (type alternative) (parent F2) (sameGroupMembers F21 F23) (requires nil) (excludes nil)))
```

(그림 3) DF1의 특성 모델에 대한 facts

3.3 오류 검증을 위한 JESS 규칙

특성 모델에 표현되어 있는 오류를 검사하기 위해서는 표1과 표2에 정의되어 있는 모든 오류들에 대하여 이를 검증하기 위한 JESS 규칙을 정의해야 한다.

JESS의 모든 규칙은 (defrule)이라는 구조체를 사용하여 정의된다. 심볼 '=>'는 if part(left-hand side)와 then part(right-hand side)를 분리한다. JESS 규칙의 동작 원리는 다음과 같다: Working memory에 존재하는 fact 집합 중 JESS 규칙의 if part와 match되는 fact가 존재한다면 이 규칙의 right part가 실행 (execute)된다.

본 논문에서는 특성 모델에 존재 가능한 오류를 검증하기 위한 모든 규칙을 JESS 규칙으로 정의하고, working memory에 존재하는 한 특성 모델에 대한 사실에 기반하여 규칙이 적용되고 오류에 대한 설명을 보여준다.

예를 들어 DF1 오류를 검증하기 위한 규칙은 그림4와 같다. 1번 라인을 보면 defrule 함수를 이용하여 새로운 규칙을 정의하며 그 이름은 checkDF1임을 알 수 있다. 7번 라인 => 이전의 문장들은 규칙의 if 부분에 해당된다. 3, 4, 5번 라인은 working

memory에 존재하는 특성 모델에 대한 fact들 중에서 DF1의 경우에 해당하는 구조를 갖는 특성 모델이 존재하는지 검사하는 부분이다. 6번 라인은 f21 특성이 f1 특성이 excludes하는 특성 리스트 중 하나의 멤버인지를 검사하는 부분이다. 이와 같이 if 부분이 working memory에 존재하는 특성 모델과 match되면 8번 라인의 then 부분이 실행된다. 8번 라인은 f21 특성이 dead feature임을 화면에 출력하는 문장이다.

본 논문의 검증 도구는 모든 dead feature 오류와 full mandatory feature 오류 검증을 위한 규칙이 정의되어 있다.

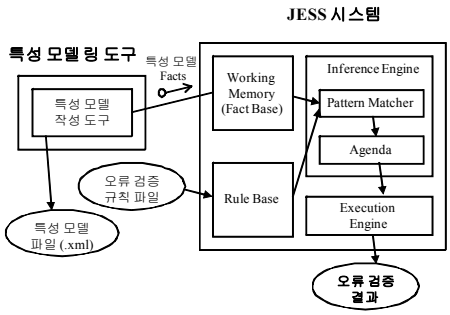
```
1:    (defrule checkDF1
2:    "DF1을 검사하는 규칙"
3:    (feature (name ?f2) (type mandatory) (parent
  ?parent2) (sameGroupMembers
  $?sameGroupMembers2) (requires $?requires2)
  (excludes $?excludes2))
4:    (feature (name ?f21) (type alternative) (parent
  ?f2) (sameGroupMembers $?sameGroupMembers21)
  (requires $?requires21) (excludes $?excludes21))
5:    (feature (name ?f1) (type mandatory) (parent
  ?parent1&~?f2) (sameGroupMembers
  $?sameGroupMembers1) (requires $?requires1)
  (excludes $?excludes1))
6:    (test (member$ ?f21 $?excludes1))
7:    =>
8:    (printout t "Error in Feature Model: " ?f21 " is
  a dead feature." crlf)    )
```

(그림 4) DF1 오류를 검증하기 위한 JESS 규칙

4. 특성 모델 검증 도구 구현

4.1 전체 아키텍처

본 논문에서 제안하는 JESS 규칙 기반 시스템을 이용한 특성 모델 검증 도구의 전체 구조는 그림5와 같다. 특성 모델 작성 도구가 생성하는 특성 모델 fact들은 working memory로 옮겨지고 rule base에 저장되어 있던 오류 검증 규칙 파일이 동작 하여 특성 모델 검증을 실행한다.



(그림 5) 특성 모델 검증 도구의 전체 구조

4.2 프로그램 실행 화면

그림6은 JESS 시스템을 활용한 특성 모델 검증 도구의 실행 화면을 보여준다. 화면의 특성 모델은 소프트웨어 제품 라인 기술들에 대한 비교 평가를 위해 [17]에서 제안한 표준 문제인 Graph Product Line(GPL) 특성 모델을 나타낸다. Search 특성이, alternative 특성인 Undirected와 requires 관계를 가지고 동시에 Unweighted 특성과 excludes 관계를 가질 때의 검증 결과를 보여준다. 화면 아래쪽이 검증 결과를 보여주며 특성 모델러가 requires 또는 excludes 관계를 생성하는 순간 검증 규칙이 동작하여 dead feature 또는 full mandatory feature가 발생했을 경우 그 사실을 알려주어 특성 모델러가 정확한 특성 모델을 생성할 수 있게 해준다.

4.3 평가

본 논문에서 제안한 JESS 프로그램에 기반을 둔 특성 구성 검증 기법은 다음과 같은 장점을 가진다.

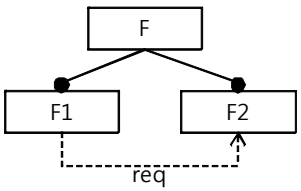
- 다른 도구와의 결합성

JESS 시스템이 가지는 자바 언어와의 결합 능력으로 인해 본 논문에서 제안한 특성 모델 검증 기법을 기존의 소프트웨어 제품 라인 지원 도구나 새로운 도구 개발 시 쉽게 통합할 수 있다. 특성 모델에 포함되어 있는 오류를 검증하는 JESS 규칙 파일은 거의 변화가 없으므로 특성 모델을 fact로 표현하는 부분만 구현하면 다른 도구에서도 쉽게 이용이 가능하다.

- 확장성

특성 모델에 대한 데이터는 fact로서 표현되고 특성 모델이 가져야 할 제한 조건은 rule로 표현되기 때문에, 특성 구성에서 검증되어야 할 새로운 규칙이 필요한 경우 이를 쉽게 추가할 수 있다. 또한. 특성 모델이 확장되거나 변경되는 경우 fact를 수정함으로써 변경 사항을 쉽게 반영할 수 있다.

예를 들어 그림7과 같은 특성 모델의 redundancy 오류를 검증하는 기능을 추가한다고 하자. 이 예제를 보면 F1과F2는 mandatory이므로 F1이 특성 구성에 포함되는 경우 F2도 반드시 특성구성에 포함되어야 하므로 requires 관계는 중복된 정보를 표현한다.



(그림 7) redundancy 오류의 한 예

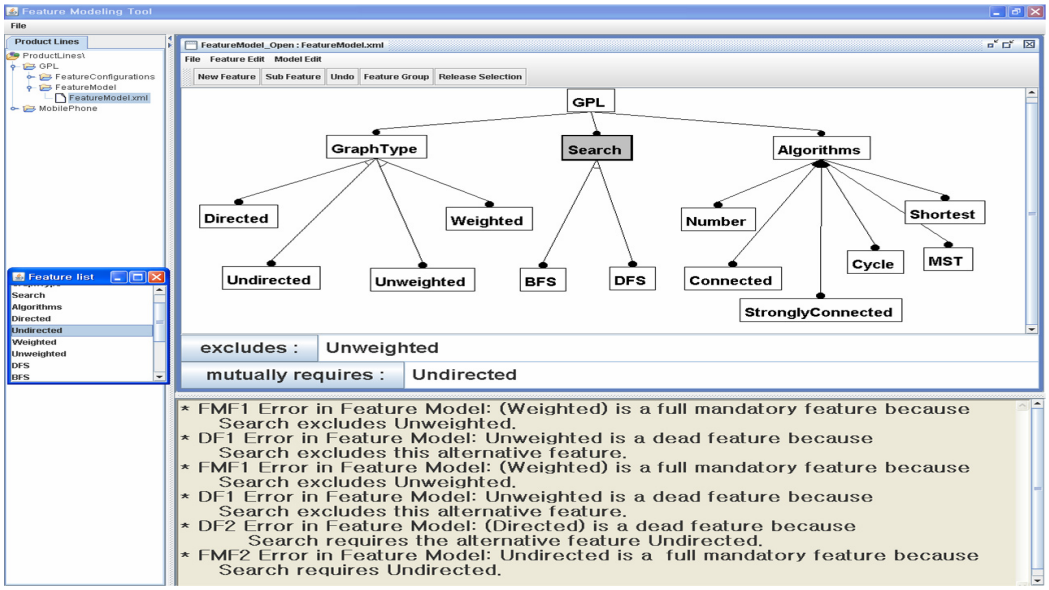
이 오류를 검증하기 위해 본 논문의 도구에서는 이 오류를 검증하기 위한 규칙을 JESS 규칙으로 정의하여 규칙 파일에 추가하기만 하면 된다. 그림 8은 그림7에 포함되어 있는 redundancy 오류를 검증하기 위한 JESS 규칙을 보여준다.

- 실시간 오류 원인 명시

특성 모델에서 requires나 excludes 관계를 설정할 때마다 JESS 규칙이 동작하므로 오류가 포함되는 상황이 발생할 때마다 오류의 종류와 이에 대한 자세한 설명이 출력되므로 특성 모델러는 오류 없는 특성 모델을 쉽게 생성할 수 있다 .

```
(defrule checkR1
  "R1을 검사하는 규칙"
  (feature (name ?f1) (type mandatory) (parent ?parent) (sameGroupMembers $?sameGroupMembers1) (requires $?requires1) (excludes $?excludes1))
  (feature (name ?f2&~?f1) (type mandatory) (parent ?parent) (sameGroupMembers $?sameGroupMembers2) (requires $?requires2) (excludes $?excludes2))
  (test (member$ ?f2 $?requires1))
  =>
  (printout t "Error in Feature Model: Redundancy is found between " ?f1 " and " ?f2 " features." crlf) )
```

(그림 8) redundancy 오류를 검증하기 위한 JESS 규칙



(그림 6) 특성 모델 검증 도구 화면

5. 결론 및 향후 연구

본 논문에서는 JESS 규칙 기반 시스템을 이용하여 특성 모델의 유효성을 검증하는 기법을 기술하고 이를 이용한 특성 모델 검증 도구를 제안하였다. 특성 모델 검증 기법에 규칙 기반 시스템을 이용한 경우는 본 논문이 최초의 시도로 사료된다.

특성 모델러가 두 특성 모델 사이의 requires 또는 excludes 관계를 생성하는 순간 JESS 규칙이 동작하여 실시간으로 특성 모델을 검증하고 오류의 종류와 그 원인에 대한 설명을 제공함으로써 특성 모델러가 오류를 포함한 특성 모델을 생성하지 않도록 도와준다.

본 논문의 기법은 새로운 오류가 정의되면 이에 대한 규칙만을 추가하면 되기 때문에 확장성이 뛰어나다. 또한 JESS 시스템이 자바 기반으로 구축되어 있기 때문에 기존의 자바 기반 소프트웨어 개발 툴과의 결합성이 좋다.

향후 연구 과제로는 시맨틱 웹 표준 언어를 지원하는 도구인 Protégé-OWL과의 결합, 보다 큰 예제 시스템으로의 적용, 컴포넌트 기반 소프트웨어 제품 라인 공학 지원 방법 등에 대한 연구가 있다.

참 고 문 헌

[1] P. Clements and L. Northrop, "Software Product Lines: Practices and Patterns", Addison Wesley, 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] D. Batory, D. Benavides, A. Ruiz-Cortes, Automated Analysis of feature models: Challenges ahead, Communications of the ACM 49 (12), pp.45-47, 2006.

[4] D Benavides, A Ruiz-Cortes, P Trinidad, S Segura, A survey on the automated analyses of feature models, Jornadas de Ingenieria del Software y Bases de Datos (JISBD), 2006.

[5] M. Mannion. Using First-Order Logic for Product Line Model Validation. In Proceedings of the Second Software Product Line Conference (SPLC2), LNCS 2379, pages 176‐187, San Diego, CA, 2002. Springer.

[6] W. Zhang, H. Zhao, and H. Mei. A propositional logic-based method for verification of feature models. In J. Davies, editor, ICFEM 2004, volume 3308, pages 115‐130. Springer‐Verlag, 2004.

[7] D. Batory. Feature models, grammars, and propositional formulas. In Software Product Lines Conference, LNCS 3714, pages 7‐20, 2005.

[8] H. Wang, Y. Li, J. Sun, H. Zhang, and J. Pan. A semantic web approach to feature modeling and verification. In Workshop on Semantic Web Enabled Software Engineering (SWESE'05), November 2005.

[9] H. H. Wang, Y. F. Li, J. Sun, H. Zhang and J. Pan. Verifying Feature Models Using OWL. In Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 5(2):117-129, June 2007.

[10] D. Benavides, A. Ruiz-Cort´es, and P. Trinidad. Automated reasoning on feature models. LNCS, Advanced Information Systems Engineering: 17th International Conference, CAiSE 2005, 3520:491‐503, 2005.

[11] P. Trinidad, D. Benavides, and A. Ruiz-Cort´es. Explanations for Agile Feature Modeling. In Proceedings of the First International Workshop on Agile Product Line Engineering (APLE'06), Baltimore, MD, USA, 2006.

[12] Czarnecki, K. and Kim, P., "Cardinality-based feature modeling and constrains: A progress report", In Proceedings of the International

Workshop on Software Factories at OOPSL 2005.

[13] Sun, J., Zhang, H., Li, Y., Wang, H., “Formal semantics and verification for feagure modeling” In Proceedings of the ICECSS05.

[14] 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.

[15] Ernest Friedman-Hill, ″JESS in Action″, Manning, 2003.

[16] T. von der Massen and H. Lichter. Deficiencies in featuremodels. In T. Mannisto and J. Bosch, editors, Workshop on Software Variability Management for Product Derivation - Towards Tool Support, 2004.

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

◐ 저 자 소 개 ◑

최 승 훈 (Seung Hoon Choi)

1990년 서울대학교 계산통계학과 졸업(학사)
1994년 서울대학교 대학원 계산통계학과 졸업(석사)
1999년 서울대학교 대학원 계산통계학과 졸업(박사)
2000년 ~ 현재 덕성여자대학교 컴퓨공학부 교수
2004년 ~ 2005년 George Mason University 방문 연구
관심분야 : 소프트웨어 프러덕트 라인, 온톨로지, 자동 생성 프로그래밍
E-mail : csh@duksung.ac.kr