韓國 컴퓨터情報學會 論文誌
第13卷 第5號, 2008. 9.

2008-13-5-1-7

온톨로지 기반 Feature 모델에서 Class 모델로의 변환 기법

김 동 리*, 송 치 양**, 강 동 수*, 백 두 권***

An Ontology - based Transformation Method from Feature Model to Class Model

Dong-Ri Kim*, Chee-Yang Song**, Dong-Su Kang*, Doo-Kwon Baik***

요 약

현재 유사 도메인에 대한 feature 모델과 class 모델간의 재사용을 위해, 모델 차원에서 상호변환 연구와 두 모델간 온톨로지를 이용한 변환 연구가 있으나, 메타모델을 통한 일관성 있는 변환이 되지 못하며, 각 모델이 가진 변환 대상 모델링 요소가 충분치 않고, 특히, 자동 변환 알고리즘 및 지원 툴을 제공하지 않음으로써 모델간 재사용의 저하를 초래하고 있다. 본 논문에서는 메타모델 상에서 온톨로지를 사용한 feature 모델을 class 모델로의 변환 방법을 제시한다. 이를 위해, feature 모델, class 모델 및 온톨로지에 대한 메타모델을 재정의하고, 각 메타모델별 모델링 요소에 대한 속성을 정의한다. 이 속성들에 기반하여 feature 모델과 온톨로지 간 그리고 온톨로지와 class 모델간의 변환 규칙 프로파일을 집합 이론과 명제논리로 정의한다. 이러한 변환의 자동화 구축을 위해 변환 알고리즘을 생성하고, 지원 툴을 구현한다. 제시한 변환 규칙 및 툴을 사용해 전자결재시스템을 통해 실제 적용한다. 기대효과로써, 기 구축된 feature 모델을 class 모델로 변환하여 상이한 개발방법간에 생성된 모델을 재사용을 할 수 있다. 특히, 온톨로지를 사용해서 의미적 변환의 모호성을 해소시킬 수 있으며, 변환의 자동화 및 모델간 일관성을 유지시켜줄 수 있다.

Abstract

At present, for reuse of similar domains between feature model and class model, researches of transformation at the model level and of transformation using ontology between two models are being made, but consistent transformation through metamodel is not made. And the factors of modeling transformation targets are not sufficient, and especially, automatic transformation algorithm and supporting tools are not provided so reuse of domains between models is not activated. This paper proposes a method of transformation from feature model to class model using ontology on the metamodel. For this, it re-establishes the metamodel of feature model, class model, and ontology, and it defines the properties of modelling factors for each metamodel. Based on the properties, it defines the profiles of transformation rules between feature model and ontology, and between ontology and class model, using set theory and propositional calculus. For automation of the transformation, it creates transformation algorithm and supporting tools. Using the proposed transformation rules and tools, real application is

• 제1저자 : 김동리 교신저자 : 백두권
• 접수일 : 2008. 7. 25, 심사일 : 2008. 8. 21, 심사완료일 : 2008. 9. 25.
* 고려대학교 컴퓨터·전파통신공학과 ** 경북대학교 소프트웨어공학과 교수
*** 고려대학교 컴퓨터·전파통신공학과 교수
※ 본 연구에 참여한 연구자는 '2단계 BK21' 사업의 지원을 받았음.

made through Electronic Approval System. Through this, it is possible to transform from the existing constructed feature model to the class model and to use it again for a different development method. Especially, it is possible to remove ambiguity of semantic transformation using ontology, and automation of transformation maintains consistence between models.

▸ Keyword : 휘처모델(feature model), 클래스모델(class model), 모델변환(model transformation), 메타모델(metamodel), 온톨로지(ontology), 재사용(reuse)

Ⅰ. 서 론

제품계열공학에서는 소프트웨어의 품질과 생산성 향상을 위해, 유사한 기능을 지닌 소프트웨어 제품 개발에 있어서 기존 산출물의 재사용을 중요시 하고 있다. 소프트웨어 공학에서 재사용의 초점은 소스 코드의 재사용에서 소프트웨어 설계의 재사용으로, 설계 재사용에서 다시 도메인 공학에 기반을 둔 재사용으로 그 패러다임이 옮겨져 왔다. 이러한 변화는 소스 코드의 재사용을 위해서는 설계의 재사용이 선행되어야 했고, 또한 설계의 재사용을 하기 위해서는 도메인 공학을 통해서 공통적으로 쓰이는 도메인 자산을 먼저 개발해야 했기 때문이다. 따라서 소프트웨어 재사용 연구는 도메인 분석 및 공학에 중점을 두고 수행되어 왔다[1,2].

재사용에 초점을 둔 개발방법으로 feature 기반의 FODA와 FORM의 개발방법이 있고, 객체 지향의 UML을 이용한 PLUS 개발 방법이 있다[2-7]. 이 방법들은 그들의 개발방법론 내에서 산출물들을 재사용 할 수 있다. 나아가, 상이한 모델링 방법간의 재사용에 대한 연구가 있어왔다. 즉, feature 모델의 class 모델로의 변환 그리고 두 모델을 온톨로지로 변환하는 기법들이 있다. 먼저, feature 모델과 class 모델간의 직접적 변환 연구[8]는 모델 아키텍처 계층 구조상의 모델 차원에서 모델이 구성하는 모델링 요소를 상호 맵핑함으로써 모델간 변환을 제시했다. 그러나 이들은 변환시 각 모델이 지닌 모델링 요소를 충분히 제공하지 못하였으며, 의미적 변환을 고려치 않았고, 메타모델(Metamodel) 차원에서 변환 맵핑을 접근하지 않아 모델의 확장 및 변경에 유연한 접근이 어렵다. 한편, 온톨로지를 사용하여 의미적 변환의 손실을 최소화하기 위한 변환방법으로 feature 모델에서 Ontology로, 다시 Ontology에서 class 모델로의 접근 연구는 전무하며, 다만 부분적으로 feature 모델을 Ontology로의 변환 그리고 Ontology를 class 모델로의 변환에 대해 약간의 연구가 있다[9-12]. 그러나 메타모델을 통한 체계적이고 일관성 있는 변환이 되지 못하고, 변환 대상 모델링 요소가 충분치 않으며, 특히 자동 변환 알고리즘 및 지원 툴을 제공하지 않음으로써 상이한 개발방법(feature 기반 개발방법, 객체기반 개발방법)에 의해 구축된 모델의 재사용 저하를 초래하고 있다.

이에, 본 논문에서는 메타모델 기반의 온톨로지를 사용한 feature 모델에서 class 모델로의 일원화된 변환 방법을 제시한다. 이를 위해, 기존 연구에서 제시된 feature 모델, class 모델 및 ontology에 대한 메타모델을 수정하여 재정립하고, 각 메타모델별 모델링 요소에 대한 속성을 정의한다. 이 속성들에 기반하여 feature 모델과 ontology간 그리고 ontology와 class 모델간의 변환 규칙 프로파일을 정하고 이를 명확히 표현하기 위해 집합 이론과 명제논리로 정의한다. 이러한 변환의 자동화를 위해 변환 알고리즘을 생성하고, 지원 툴을 구현한다. 제시한 변환 규칙, 과정 및 툴을 사용해서 기 구축된 전자결재시스템 모델을 변환하여 그 실효성을 보인다. 이로써, 기 구축된 feature 모델을 class 모델로 변환하여 상이한 개발방법간에 재사용을 할 수 있다. 또한, 온톨로지의 OWL을 사용하여 웹상에서 두 모델간 변환을 제공한다. 특히, 온톨로지를 사용하면 의미적 변환의 모호성을 해소시킬 수 있으며, 변환의 자동화로 모델간 일관성을 유지시켜 줄 수 있다.

본 논문의 구성은 다음과 같다. 제 2장은 관련연구에 대해 설명하고, 제 3장은 본 논문의 feature 모델에서 class 모델로의 변환 프로세스를 제시한다. 제 4장은 feature 모델, 온톨로지, class 모델의 속성을 분석하여 feature와 class 모델간의 속성변환 방법, feature 모델 - Ontology - class 모델의 변환 알고리즘에 대해 설명하고, 제 5장은 모델간 변환 툴의 구현을 다루며, 제 6장은 본 논문에서 제안한 접근방법의 효과성을 보이기 위해 전자결제 시스템 도메인에 대한 case study를 보여주며, 제 7장은 본 논문에서 제시한 방법과 기존 연구방법을 비교평가 한다.

Ⅱ. 관련연구

제품계열공학에서 feature와 class는 도메인 분석에 있어 중요한 요소로, 이들을 명시적으로 표현하고, 이들 모델을 재

사용하기 위한 연구가 진행 되었다. 이장에서는 기존에 연구되어 온 제품계열공학 개발방법과 이들 방법에 의해 생성된 모델들간의 변환에 관한 연구에 대해 살펴본다.

2.1 제품계열공학 개발방법

2.1.1 Feature를 이용한 개발방법

feature를 이용한 대표적인 개발방법으로 FODA (Feature-Oriented Domain Analysis)가 있다. 이 방법은 시스템의 집합 중에서 주도적이고 독특한 feature를 식별하여, 이것들을 바탕으로 도메인을 분석한다. [4]에서 feature는 "구현, 테스트되고 배포 및 유지 되어야하는 기능적/비기능적 추상화를 말하는 것으로 요구사항이나 특징적인 기능"이라 정의한다. 또, feature 모델의 재사용에 초점을 맞춘 방법에는 FORM(Feature-Oriented Reused Method)이 있다 [2]. FORM의 핵심은 도메인의 feature를 분석하고, 재사용 가능한 도메인 제품 개발시 해당 feature를 사용하는데 있다. 재사용을 위한 첫 단계로 FODA에서 정의한 4가지 계층인 능력, 운용환경, 도메인 기술, 구현기술 상에서 하나의 특별한 도메인 제품의 공통성을 분석한다. feature 모델을 바탕으로 AND/OR 그래프를 이용하여 공통성을 추출하는데, AND 노드는 feature의 필수적인 feature를 말하고, OR 노드는 다른 제품으로 선택 될 수 있는 양자택일의 feature를 말한다.

2.1.2 객체지향 개발방법

UML을 이용한 객체지향 개발 방법으로 PLUS (Product Line UML based Software Engineering)가 있다[7]. 이 방법은 시스템 개발에 있어서 UML 기반 모델링 기법을 확장하여, 제품계열공학 개발을 지원하기 위해 여러가지 개념과 기법을 제공한다. 즉, 프로덕트 라인에 존재하는 제품들 사이의 공통점과 차이점을 분석하여 초기 단계부터 명확하게 모델링을 하기 위한 기법이다. PLUS는 크게 3단계로 구성되어 있는데 1단계인 요구사항 모델링에서는 use case 모델링을 사용하여 공통점과 차이점을 모델링하고, 2단계는 분석 모델링 단계로 정적구조, 동적 상호작용, 동적 상태 기계 및 특성/클래스 의존성 모델링을 한다. 그리고 3단계는 설계 모델링 단계로 소프트웨어 아키텍처 패턴을 결정하고, 컴포넌트 기반 소프트웨어 설계를 한다.

2.2 모델간 변환 기법

2.2.1 Feature 모델과 Class 모델간 변환

feature 모델을 class 모델로 변환시키는 방법에는 feature diagram을 이용한 UML diagram을 생성하는 연구가 있다[8]. 이 방법은 feature 모델과 이 feature 모델을 이용해서 만든 UML모델에서 각 요소들을 직접 맵핑시켜, feature 모델의 속성인 mandatory, alternative, optional에 따라 선택적으로 UML모델을 만드는 방법이다. 즉, feature 모델과 class 모델의 추상화 정도를 다른 레벨로 가정하고, feature와 class를 1:n으로 맵핑을 하였다. 하지만, 이 방법은 모델 차원에서 개별 feature를 class 모델 요소와 직접 맵핑을 시켜야 하기 때문에 맵핑해야 할 요소들이 많고, 또 개발자 직관에 의한 개별 수동 맵핑으로 맵핑시 많은 오류를 내포하고 있어 다른 도메인에서 feature 모델을 재사용하기 어렵다.

2.2.2 Feature 모델과 Ontology간 변환

feature 모델을 OWL로 변환하는 연구로는 [9,12]이 있다. 그들의 연구는 feature의 속성을 분석하여 온톨로지를 이용하여 표현하는 방법들이다. 그들 연구에서 feature 모델의 행위(action)와 용어(term)를 owl:class로 정의하고, 패싯(facet)은 owl:Property로 정의하며, 행위와 용어 사이의 관계를 subclassOf로 표현하고 있다. 그러나 feature의 속성과 관계를 표현하는데 있어서 단순하게 owl:Property와 subclassOf로 표현하고 있어, feature의 세부속성(식별형, 선택형, 관리형, 관계형)과 feature들간의 여러 관계의 속성들을 표현하지 못한다. 따라서 이 방법은 완전하지 못한 변환을 가진다.

2.2.3 Ontology와 Class 모델간 변환

온톨로지를 표현하는 언어로서 OWL은 표현력이 좋고, 의미 손실이 없으며, 호환성 및 정확성이 우수한 언어이다. OWL을 class 모델로 변환하는 연구에는 [10,11,13]이 있다. 이 연구들은 OWL의 속성과 class 속성을 분석하여, 이들 속성들 간의 맵핑으로 모델 변환을 하는 방법이다. 예로, OWL의 속성인 owl:class를 class 모델의 class로, owl:DatatypeProperty를 attribute로, owl:objectProperty, owl:onProperty, owl:cardinality, owl:restriction을 Association으로, owl:subclassOf를 Generalization으로 각각 맵핑하여 표현을 한다. 그러나 OWL의 UnionOf 등 맵핑되지 못한 표현들이 있어, 결국 feature 모델을 class 모델로 변환하기 위해서는 OWL과 class 모델에 대한 좀 더 상세한 분석과 정의가 필요하다.

III. Feature 모델에서 Class 모델로의 변환 프로세스

본 장에서는 상이한 개발방법에 의해 생성된 feature 모델을 class 모델로 변환하기 프로세스 모형을 제안한다.

본 논문의 목적은 상이한 개발방법에 의해 모델링되는 feature 모델을 class 모델로 변환하여 기 산출된 설계 모델의 재사용을 제공하는 것이다. feature 모델과 class 모델은 특정 도메인에 대한 공통적 지식 혹은 정보의 구조를 표현하는 것이다. feature 모델은 기능적(서비스) 및 비기능적 특성 등을 표현하는데 비해, class 모델은 기능적 특성을 나타낸다. 그래서 두 모델의 공통적 표현요소인 기능적 특성을 대상으로 표현하면 모델간 변환을 수행 할 수 있다. 또한, feature 모델은 개발 단계 수준상 상위 수준에 해당한다. 반면, class 모델은 그 추상화 정도에 따라 개념적 단계, 명세적 단계, 상세적 단계로 나타낼 수 있다. 그래서 본 논문에서는 feature 모델과 추상화 수준이 일치하도록 상위 수준의 class 모델로 변환토록 한다. 즉, feature 모델의 서비스 부분을 기반으로 class, attribute 그리고 class간 관계 정도를 class 모델로 변환한다.

아울러, feature 모델과 class 모델을 1:1로 맵핑함을 전제로 한다. 물론, 중·대규모 시스템인 경우, 한 개의 feature 모델은 영역별로 다수의 class 모델들을 생성 할 수 있다. 그러나 전체 도메인을 대상으로 하나 여러 서브 도메인을 대상으로 하나 메타모델 차원에서 feature 모델을 class 모델로 변환하기에 접근방법은 동일하므로 두 모델간의 맵핑을 1:1로 가정한다.

본 연구에서는 feature 모델에서 class 모델로의 변환을 위해 온톨로지를 사용한다. 변환을 위해 온톨로지를 사용하는 이유는 기본적으로 온톨로지는 공통적 이해를 공유토록 명세를 하고, 또한 두 모델의 특성을 다양한 정형적 언어와 OWL 표현 언어(Description Logic)를 가지고 잘 표현 할 수 있기 때문이다. 즉, 두 모델의 변환시 의미적 손실을 제거할 수 있다. 또한, 정보를 표현하는데 있어서 feature 모델은 계층적 트리구조의 형식으로 나타내고, class 모델은 네트워크형 구조로 나타낸다. 따라서 정보 표현의 구조적 상이성 때문에 feature 모델을 class 모델로 직접 맵핑하는 데에는 제약이 따른다. 하지만, ontology는 트리형 및 네트워크형 구조의 표현을 모두 지원하기에 feature 모델과 class 모델간 변환시 구조적 상이성을 해결 할 수 있다. 그리고 부수적으로 웹을 통해 생성된 모델을 공유 할 수 있으며, 일관성 있는 변환의 자동화 실현을 도모 할 수 있기 때문에 온톨로지를 사용한다.

그림 1에서와 같이 Feature 모델에서 Class 모델로의 변환 프로세스는 크게 2단계로 이루진다.

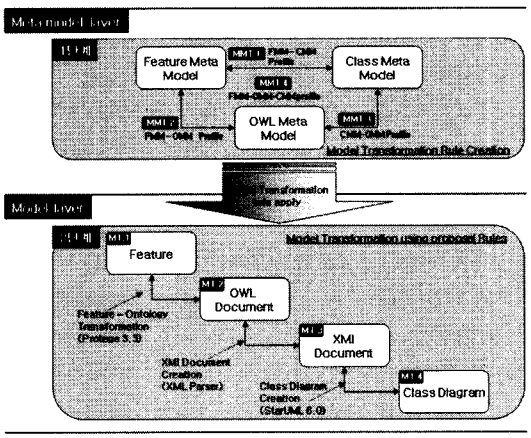

그림 1. Feature 모델에서 Class 모델로의 변환 프로세스
Fig. 1 Transformation process from feature model to class model

1단계에서는 모델 변환 규칙을 생성하는 단계로 메타모델 차원에서 각 모델간의 속성을 정의하고, 그 속성간의 관계를 정의하는 단계로 세부 프로세스는 다음과 같다.

(1) MMT_1 단계: Feature 메타모델과 Class 메타모델간의 속성을 정의하고 맵핑 규칙을 생성
(2) MMT_2 단계: Feature 메타모델과 OWL 메타모델간의 속성을 정의하고 맵핑 규칙 생성
(3) MMT_3 단계: Class 메타모델과 OWL 메타모델간의 속성을 정의하고 맵핑 규칙 생성
(4) MMT_4 단계: MMT_1 단계에서 MMT_3 단계까지 만들어진 룰을 이용하여 Feature 메타모델, OWL 메타모델, Class 메타모델의 속성을 정의하고 세 모델간의 맵핑 규칙을 정립

이 단계를 통해 모델 차원에서 사용할 Feature 모델에서 class 모델로의 변환 프로파일이 완성되며, 세부 단계는 제 4장의 단원 4.2에서 정의한다.

2단계에서는 1단계에서 생성된 규칙을 이용하여 실제 모델 차원에서 특정 응용 모델 구축으로써 세부 단계는 다음과 같다.

(1) MT_1 단계: feature 모델을 생성하거나 재사용할 기존의 feature 모델을 선정
(2) MT_2 단계: MT_1에서 만들어진(선택된) feature 모델을 OWL Document로 변환
(3) MT_3 단계: MT_2에서 생성된 OWL Document를 XMI Document로 변환
(4) MT_4 단계: MT_3에서 만들어진 XMI Document를 Class 모델로 변환

본 기법의 2단계에 대해서는 제 5장에서 변환 Tool로 구현되며, 제 6장 사례적용에서 그 실례를 구체적으로 보인다.

IV. Feature 모델에서 Class 모델로의 속성 변환

본 장에서는 feature 모델과 class 모델간 변환 방법을 기술하기 위해, feature 모델, OWL 모델, class 모델에 대한 메타모델과 속성을 정의하고, 그림 1의 1단계에 대한 메타모델 차원의 변환 프로파일을 정의한다.

4.1 메타모델 및 속성 정의

4.1.1 Feature 메타모델

[5]의 연구에서 feature 모델의 상위 수준 카테고리 구성은 능력, 운용환경, 도메인 기술, 구현기술로 분류하고, 정의는 다음과 같다.

- 능력: 사용자 관점에서 어플리케이션의 능력
  ex) 전화기 도메인에서 전화연결 및 다이얼링, 성능
- 운용환경: 어플리케이션이 운용되는 환경
  ex) 하드웨어, 소프트웨어 플랫폼, 운영체제
- 도메인 기술: 하나의 도메인에서 공통적으로 사용되는 기술
  ex) 항공전자 도메인에서 내비게이션 기술
- 구현 기술: 알고리즘과 데이터 구조에서 설계자가 결정하는 구현기술
  ex) 동기화 메커니즘

그림 2는 [5]에서 제시한 feature 모델을 재정립하여 상위 수준 카테고리를 계층별로 class diagram으로 나타낸 feature 메타모델이다. 그림에서 범례로 "•"는 각 상위 수준 카테고리에서 세부적으로 분석 될 수 있는 개별 feature를 말한다.

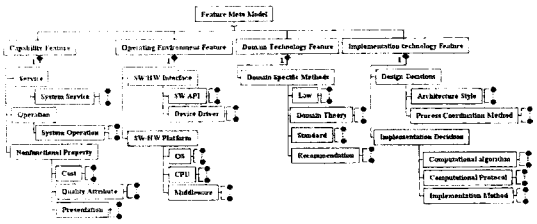

그림 2. Feature 메타모델
Fig. 2 Feature Metamodel

meta feature란 'feature의 feature'로써 feature를 표현, 기술 및 관리하기 위한 feature를 말한다. 따라서 meta feature는 제품 개발 도메인에서 사용되는 feature에 대한 공통된 이해를 가능하게 만들어 주는 유용한 정보이고, feature의 활용, 검색 및 공유를 돕기 위한 feature를 말하며, feature 모델에 내재된 feature의 내용, 구조, 의미를 체계적으로 기술하는데 사용된다. 그림 2의 feature 메타모델상의 각 feature는 표 1과 같은 세부적 속성들을 갖는다.

표 1. 개별 Feature의 속성 분류
Table. 1 Individual feature attribute classification

구 분	세부 속성	설 명
식별형 속성	Name	식별된 feature의 명시적 이름
	Feature Synset	일반적 의미에서 feature와 동일하거나 유사한 개념의 feature 이름
	Classification	도메인 내에서 부모 feature와 관계
선택형 속성	Mandatory	도메인 내의 일련의 feature들 중, 반드시 존재해야하는 feature
	Optional	도메인 내에서 일련의 feature들 중, 필수적이 아닌 선택 가능한 feature
	Alternative	도메인 내의 일련의 feature들 중, 양자택일 선택이 가능한 feature
관리형 속성	Identifier	도메인 내에서의 feature 고유한 ID Number
	Registration authority	feature를 등록한 조직, 그룹 또는 회사명
관계형 속성	Has-part	하위 feature의 관계(부분의 관계)로 composed of 또는 specialization 관계와 동치
	Part-of	상위 feature의 관계(전체의 관계)로 generalization 관계와 동치

그림 3은 표 1의 개별 feature의 속성들을 class 모델로 표현한 것이다. 개별 feature는 식별형, 선택형, 관리형, 그리고 관계형 속성으로 구분하며, 각 속성 유형별로 서브 속성들을 가진다.

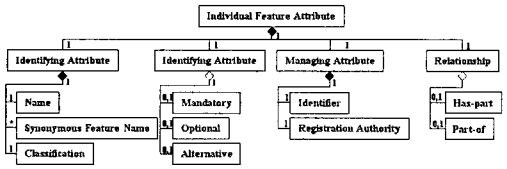

그림 3. 개별 Feature의 속성 메타모델
Fig. 3 Individual Feature attribute Metamodel

4.1.2 Class 메타모델

본 단원은 class 모델의 메타모델을 정의한다. 그림 4는 [14]의 class 메타모델을 UML 2.0에 부합하도록 수정하여 재정립 하였다.

일반 속성인 stereotype, responsibility와 관계속성인 generalization 등의 속성을 추가 하였고, 속성들의 관계와 Cardinality를 일부 수정 하였다.

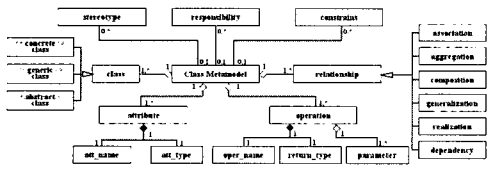

그림 4. Class 메타모델
Fig. 4 Class Metamodel

표 2는 class 메타모델의 주요 속성들을 정의한 것이다. class 모델의 속성은 class, attribute, operation 등과 같은 일반 속성 그리고 class들간의 관계를 나타내는 association, aggregation, composition 등과 같은 관계형 속성으로 분류된다.

표 2. Class 메타모델 속성 분류
Table. 2 Class Metamodel attribute classification

구 분	세부 속성	설 명
일반 속성	Class	비슷한 속성과 공통적인 행동 수단을 지닌 것들의 어떠한 범주 혹은 그룹
	Attribute	클래스에 속한 특성에 이름을 붙인 것으로 이것이 가질 수 있는 값의 범위를 설정
	Operation	객체에 요청 할 수 있는 행동
	Responsibility	특정한 타입 혹은 클래스가 해야 하는 일을 설명
	Constraints	중괄호{ } 안에 자유 형식의 텍스트가 들어 있는 형태로써, 클래스가 따라야 하는 규칙
	Stereotype	〈〈 〉〉을 사용하여 확장된 영역을 표시
관계형 속성	Association	두 클래스간의 관계
	Aggregation	전체와 부분의 관계
	Composition	Aggregation의 특수한 경우, 강한 소유의 표시
	Generalization	일반화된 사물과 특수화된 사물
	Realization	정의와 구현관계
	Dependency	의존형 관계

4.1.3 OWL 메타모델

OWL은 Web Ontology Language로써, Ontology를 표현하기 위한 대표적인 언어이다. OWL에 대한 메타모델을 정의한 것이 그림 5이다. 이것은 [15]에서 제시한 OWL의 속성을 기반으로 그 구문적 구조를 표현한 것이다.

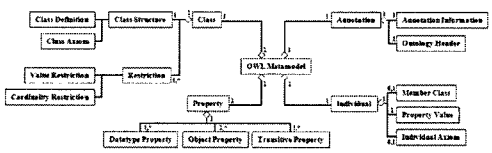

그림 5. OWL 메타모델
Fig. 5 OWL Metamodel

OWL의 속성은 크게 온톨로지의 구조나 제약을 나타내는 class, 속성들을 나타내는 property, 개별적 정의에 사용하는 individual, 그리고 온톨로지 정보를 나타내는 annotation으로 구성된다.

표 3은 그림 5의 OWL의 메타모델 요소들을 분류하여 주요 속성들을 분류, 정의하였다. 여기서 정의된 속성들은 feature 모델을 온톨로지로 표현하고, class 모델로의 변환을 위한 중간 매개체의 역할을 한다.

표 3. OWL 메타모델 속성 분류
Table. 3 OWL Metamodel attribute classification

구 분	세부 속성	설 명
Class	Class Structure	온톨로지 클래스 구조적 표현
	Restriction	온톨로지의 제약조건
Property	Datatype Property	데이터 형식 속성(속성의 값이 XML 스키마 형식이나 리터럴)
	Object Property	개체 형식 속성(속성의 값이 클래스 또는 인스턴스)
	Transitive Property	추이속성, P(x,y) and P(y,z) 이면 P(x,z) 성립
Individual	Member Class	클래스로 표현되는 각각 인스턴트들의 그룹
	Property Value	클래스로 표현되는 각각 인스턴트들의 값
	Individual Axiom	개체들 사이에 사용되는 공리
Annotation	Ontology Header	온톨로지 자체에 대한 정보
	Annotation Information	주석에 대한 정보

4.2 Feature 모델 - Ontology - Class 모델간의 변환규칙 및 프로파일

3장에서 제시한 Metamodel Layer에서의 MMT_1 단계부터 MMT_4 단계를 세부적으로 적용하여, Feature 모델 - OWL - Class 모델간의 속성 맵핑 프로파일을 생성한다.

4.2.1 변환 규칙

feature 모델, OWL, class 모델간 속성 맵핑은 단원 4.1에서 분석된 메타모델 속성들을 기반으로 각 속성들의 의미분석을 통해 이루어진다. feature 모델과 class 모델 맵핑에 있어서 동일한 속성, 유사한 속성, 상이한 속성은 다음과 같은 규칙을 따른다.

$$F=\{F_{eature}P_{roperty}\}=\{F_1,\cdots,F_n\}$$

$$C=\{C_{lass}P_{roperty}\}=\{F_1,\cdots,F_n\}$$

$$\bigcup_{i=1}^{n}F_i=\bigcup_{i=1}^{n}\{F_i\,|\,f_{syntax},\ f_{semantics}\}$$

$$\bigcup_{j=1}^{m}C_j=\bigcup_{j=1}^{m}\{C_i\,|\,C_{syntax},\ C_{semantics}\}$$

F는 feature 속성들의 집합을 나타내고, 개별 Feature Property(Fi)는 feature의 문법을 나타내는 syntax와 의미를 나타내는 semantics로 구성된다. 그리고 C는 class속성들의 집합을 나타내고 개별 Class Property(Cj)는 class의 문법을 나타내는 syntax와 의미를 나타내는 semantics로 구성된다.

· 동일한 속성

동일한 속성은 F집합과 C집합의 교집합 중에 feature와 class의 syntax와 semantics가 동일한 것을 말한다.

SameProperty

$= (F\wedge C), f_{syntax}=c_{syntax}, f_{semantics}=c_{semantics}$

feature 모델의 속성 중 식별형, 선택형, 관리형 속성들은 feature의 식별, 상태 그리고 관리를 위한 요소들로, class 모델의 class name과 속성들의 특성을 나타내는 attribute와 syntax 그리고 semantics 차원에서 같으므로, 서로 동일시 맵핑한다.

· 유사한 속성

유사한 속성은 F집합과 C집합의 교집합중, semantics는 같지만, syntax가 다른 것을 말한다.

SimilarProperty

$= (F\wedge C), f_{syntax}\neq c_{syntax}, f_{semantics}=c_{semantics}$

feature의 관계형 속성인 part-of와 has-part는 feature들의 부모-자식의 관계를 나타내는 속성으로 class 모델의 관계형 속성중 일반화(generalization), 집합(aggregation)과 semantics 차원에서 의미가 같지만, syntax 차원의 문법이 달라 유사속성으로 분류된다. 맵핑시는 속성의 의미가 같으므로 동일한 속성 맵핑시와 같이 1대1일 맵핑을 한다.

· 상이한 속성

상이한 속성은 F집합과 C집합의 교집합 원소가 아니고, semantics가 다른 속성들을 말한다.

DifferentProperty $= \neg(F\wedge C), f_{semantics}\neq c_{semantics}$

class 모델의 상이한 속성으로 responsibility, constraints, stereotype, dependency 등이 있다. 이 요소들은 추후 연구에서 다룰 것이다.

한편 OWL은 feature와 class 모델을 변환 시켜주는 중간매개체로, feature 모델과 class 모델의 요소간 의미를 명확히 하여 변환의 정확성을 재고 시켜준다. 그리고 이러한 규칙을 적용해서 feature-class, feature-ontology 그리고, ontology-class간 변환 프로파일을 정립한다.

4.2.2 Feature 모델에서 Class 모델로의 변환 프로파일

표 4는 feature 모델과 class 모델 속성을 비교하여 변환 프로파일을 나타낸 것으로 그림 1의 MMT_1 단계에 해당한다.

표 4. Feature 모델과 Class 모델간 변환 프로파일
Table. 4 Transformation profiles between Feature model and Class model

Feature model		Class model
식별형 속성	Name	class
	Feature Synset	attribute
	classification	attribute
선택형 속성	mandatory	attribute
	selective	attribute
	alternative	attribute
관리형 속성	identifier	attribute
	Registration authority	attribute
관계형 속성	Part-of	generalization
	Has-part	aggregation

4.2.3 Feature 모델에서 OWL로의 변환 프로파일

feature 모델을 class 모델로 자동 변환하고, feature의 의미를 명확히 표현하기 위해서 feature를 온톨로지로 표현한다.여기에서는 OWL을 이용하였다. Feature에서 Ontology로의 변환 프로파일을 표 5에 나타내고 있으며, 3장의 그림 1상의 MMT_2 단계를 위한 것이다.

표 5. Feature 모델과 OWL간 변환 프로파일
Table. 5 Transformation profiles from Feature model to OWL

Feature model		OWL
식별형 속성	Name	owl:class
	Feature Synset	owl:DatatypeProperty
	classification	owl:DatatypeProperty
선택형 속성	mandatory	owl:DatatypeProperty
	selective	owl:DatatypeProperty
	alternative	owl:DatatypeProperty
관리형 속성	identifier	owl:DatatypeProperty
	Registration authority	owl:DatatypeProperty
관계형 속성	Part-of	owl:subclassOf
	Has-part	owl:UnionOf

4.2.4 OWL에서 Class 모델로의 변환 프로파일

OWL-Class 모델간의 변환 프로파일을 표 6에 나타내었으며, 이 변환 프로파일은 그림 1상의 MMT_3 단계의 과정에 적용된다.

표 6. OWL과 Class 모델간 변환 프로파일
Table. 6 Transformation profiles from OWL to Class model

OWL	Class Model
owl:class	class
owl:DatatypeProperty	attribute
owl:objectProperty	association
owl:onProperty	association
owl:cardinality	multiplicity
owl:restriction	constraint
owl:subclassOf	generalization
owl:UnionOf	aggregation

4.2.5 Feature 모델-OWL-Class 모델간의 변환 프로파일

Feature모델-OWL-Class 모델간 변환 프로파일은 그림 1상의 MMT_4 단계를 위한 것이다. 이것은 이전 단원 4.2.1에서 단원 4.2.4에 이르기까지 정의한 변환 프로파일을 기반으로 표 7과 같이 전체 변환 맵핑을 정립하였다.

표 7. Feature 모델-OWL-Class 모델간 변환 프로파일
Table. 7 Transformation profiles Feature model - OWL - Class model

Feature model		OWL	Class Model
식별형 속성	Name	owl:class	class
	Feature Synset	owl:DatatypeProperty	attribute
	classification	owl:DatatypeProperty	attribute
선택형 속성	mandatory	owl:DatatypeProperty	attribute
	selective	owl:DatatypeProperty	attribute
	alternative	owl:DatatypeProperty	attribute
관리형 속성	identifier	owl:DatatypeProperty	attribute
	Registration authority	owl:DatatypeProperty	attribute
관계형 속성	Part-of	owl:subclassOf	generalization
	Has-part	owl:UnionOf	aggregation

feature 모델의 SameProperty인 식별형, 선택형, 관리형 속성은 OWL의 owl:class, owl:DatatypeProperty로 변형되고, 이것들은 다시 class 모델의 class, attribute 맵핑된다. 또한 feature 모델의 SimilarProperty인 관리형 속성은 OWL의 owl:subclasssOf, owl:UnionOf로 변환되고, 다시 feature 모델의 generalization과 aggregation으로 맵핑된다. 예로 feature 모델의 SameProperty인 mandatory는 owl의

```
〈owl:DatatypeProperty rdf:ID="selectiveValue"〉
 〈rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema
 #string"〉Mandatory〈/rdf:first〉
 〈/owl:DatatypeProperty〉
```

로 표현되고, class 모델의 attribute 속성으로 selective Value:String=Mandatory로 변환된다. 또 SimilarProperty인 Part-of는 OWL의

```
〈owl:Class rdf:ID="InteriorApproval"〉
  〈rdfs:subClassOf〉〈owl:Class rdf:ID="DocumentForm" /〉
    〈/rdfs:subClassOf〉
〈/owl:Class〉
```

로 나타내고, class 모델의 generalization의 관계로 맵핑된다.

4.3 모델간 변환의 정형명세 및 알고리즘

4.3.1 Feature 모델에서 Class 모델로의 변환 정형명세

단원 4.2에서 Feature-OWL-Class 모델간 변환 프로파일을 제시하였지만, 이 변환 관계를 수학적으로 명확히 표현하지 못하였다. 그래서 Set Theory를 이용하여 그림 6과 같이 feature 모델을 class 모델로의 변환을 표현하였다.

Step 1: Feature model → OWL Transformation

F = { {I}, {S}, {M}, {R} }
{I} = {name, featuresynset, classification | name = owl:class, featuresynset = owl:DatatypeProperty, classification = owl:DatatypeProperty}
{S} = {mandatory, optional, alternative | mandatory = owl:DatatypeProperty, optional = owl:DatatypeProperty, alternative = owl:DatatypeProperty}
{M} = {identifier, registration | identifier = owl:DatatypeProperty, registration = owl:DatatypeProperty}
{R} = {part-of, has-part | part-of = owl:subclassOf, has-part = owl:UnionOf}

Step 2: OWL → Class model Transformation

O = {owl:class, owl:DatatypeProperty, owl:subclassOf, owl:UnionOf}
C = {class, attribute, generalization, aggregation | class = owl:class, attribute = owl:DatatypeProperty, generalization = owl:subclassOf, aggregation = owl:UnionOf}

F : feature 속성들의 집합
I : feature 식별형 속성(identifying property)집합의 원소
S : feature 선택형 속성(selective property)집합의 원소
M : feature 관리형 속성(managing property)집합의 원소
R : feature 관계형 속성(relation property)집합의 원소
O : OWL 속성들의 집합
C : class 모델 속성들의 집합

그림 6. Feature 모델-OWL-Class 모델 변환의 집합이론 표현
Fig. 6 Set theory expression of model transformation among Feature model, OWL, and Class model

Step 1에서 feature 모델과 OWL의 관계를 표현하였고, Step 2에서는 OWL과 class 모델의 관계를 나타낸다.

그림 8은 본 논문에서 제안하는 Feature-OWL-Class 모델 변환을 명제논리로 표현한 것이다. 그림 6의 Set Theory는 제안한 기법을 집합관계로 표현하였지만, 모델 변환의 당위성을 나타 낼 수 없기에 수학 명제논리를 이용하여 모델 변환의 일치성을 검증한다.

- $\forall FP = \bigcup_{i=1}^{n} FP_i = \bigcup_{i=1}^{n} I_i + S_i + M_i + R_i$
- $\bigcup_{i=1}^{n} FP_i \rightarrow \bigcup_{j=1}^{\exists} OP_j$
- $\bigcup_{j=1}^{\exists} OP_j \rightarrow \bigcup_{k=1}^{\exists} CP_k$
- $\therefore \bigcup_{i=1}^{n} FP_i \rightarrow \bigcup_{k=1}^{\exists} CP_k$
- 단: $\bigcup_{k=1}^{\exists} CP_k \neq \bigcup_{i=1}^{n} FP_i$

〈범례〉
FP : Feature Property
OP: Ontology Property
CP: Class Property
I : Identifying property
S: Selective property
M: Managing property
R : Relation property

그림 7. Feature 모델-OWL-Class 모델 변환의 명제논리 표현
Fig. 7 Proposition logical expression of model transformation among Feature model, OWL, and Class model

그림 7에서, Feature Property는 n개의 Feature Property로 나타낼 수 있고, 이것은 다시 n개의 식별형 속성, 선택형 속성, 관리형 속성, 관계형 속성들의 집합으로 나타낼 수 있다. 그리고 이 n개의 Feature Property는 Ontology Property들의 합으로 나타낼 수 있고, 이 Ontology Property는 다시 Class Property로 나타 낼 수 있다. 즉, n개의 Feature Property는 Class Property로 나타낼 수 있다. 그리고 역으로 모든 Class Property가 feature property로 표현 되는 것은 아니다 라는 것을 보여준다.

4.3.2 Feature 모델에서 Class 모델로의 변환 알고리즘

단원 4.1에서 feature, class, OWL에 대한 메타모델을 정의하고, 단원 4.2에서 정의된 메타모델을 이용하여 모델들간의 변환 프로파일을 생성하였다. 본 단원에서는 정의된 변환 프로파일을 이용하여 Feature 모델-OWL-Class 모델간 변환 알고리즘을 제시한다. 그림 8은 Feature 모델-OWL-Class 모델간 변환 알고리즘을 Pseudo Code로 표현한 것이다.

```
Function name : TransformaitonFeatureToClass

In. : FP, OP // FP(Feature Porperty) OP(Ontology Property)
Out. : CP // CP(Class Property)

Step 1 // Changing Feature to Ontology
  Switch (FP) {
   if (FP = identifyingProperty) {
    case 'name' : OP='owl:class';
    case 'featureSynet' : OP='owl:DataProperty';
    case 'classification' : OP='owl:DataProperty';
   }else if (FP = selectiveProperty) {
    case 'mandatory' : OP='owl:Dataproperty';
    case 'optional' : OP='owl:DataProperty';
    case 'alternative' : OP='owl:DataProperty';
   }else if (FP = managingPorperty) {
    case 'identifier' : OP='owl:Dataproperty';
    case 'registationAuthority' : OP='owl:DataProperty';
   }else if (FP = relationPorperty) {
    case 'part_of' : OP='owl:SubclassOf';
    case 'has_part' : OP='owl:UnionOf';
    }
  }

Step 2 // Changing Ontology to Class
  Switch (OP) {
   if (OP = OWLclass) {
    case 'owl:class' : CP='CD:class';
   }else if (OP = OWLproperty) {
    case 'owl:Dataproperty' : CP='CD:attribute';
   }else if (OP = OWLclassStructure) {
    case 'owl:subClassOf' : CP='CD:generalization';
    case 'owl:UnionOf' : CP='CD:aggregation';
   }
  }
```

그림 8. Feature 모델-OWL-Class 모델간 변환 알고리즘
Fig. 8 Transformation algorithm among Feature model, OWL, and Class model

모델 변환의 알고리즘을 정립함으로써, 변환의 일관성을 유지할 수 있고, 또한 알고리즘을 이용한 모델 변환 자동화 CASE Tool을 구현할 수 있다.

Step 1에서는 feature 속성을 입력 받아, 이것을 Ontology 속성으로 변환하는 과정을 표현한 것이다.

Step 2에서는 변환된 Ontology 속성을 입력받아, 이것을 class 속성으로 변환 시킨다. ontology 속성은 크게 3가지 (owlClass , owlProperty, owlStructure)로 구분되고, 이것은 다시 class 모델의 세부 속성들과 맵핑된다.

V. Feature-to-XMI 툴 구현

본장에서는 제시한 Feature 모델-OWL-Class 모델간의 변환 알고리즘을 자동화하기 위한 툴 구현에 대해 설명한다. 모델 변환은 3장의 2단계에서 제시한 프로세스를 따른다. MT_1 단계와 MT_2 단계는 Feature to OWL 변환 룰을 적용하고, Protege 3.3을 이용하여 온톨로지로 변환을 한다. MT_3 단계는 구현된 툴을 이용하여 온톨로지를 XMI Document 변환 시킨다. 그리고 MT_4 단계는 기존 상용 툴인 StarUML 6.0을 이용하여 class 모델을 생성하였다.

본 논문에서는 그림 9상에서 MT_3 단계의 온톨로지를 XMI로 변형 시키는 툴을 개발 하였다. 개발 환경은 다음과 같다.

개발 플랫폼: Window XP, CPU Pentium 3GHz
개발 언어(Programming Language): JSP, Java DOM
OWL / UML툴: Protege 3.3 / StarUML 6.0
사용자 인터페이스(user interface): 웹 기반 어플리케이션

그림 9는 온톨로지를 XMI로 변환하기 위한 프로세스를 나타낸 것이다.

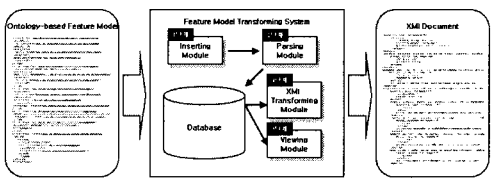

그림 9. Ontology에서 XMI로 변환 프로세스
Fig. 9 Transformation process from Ontology to XMI

Feature Model Transforming System은 다음과 같은 절차를 따른다.

(1) 1 단계: Feature 모델 - OWL 변환 프로파일을 이용하여 온톨로지로 명세된 feature 모델 파일을 로딩한다.
(2) 2 단계: Java Dom을 이용하여 불러들인 feature 모델 파일을 각 요소별 Parsing 하여 DB에 저장한다.
(3) 3 단계: View 모듈을 이용하여 DB에 저장된 feature 요소를 본다.
(4) 4 단계: DB에 저장된 feature 요소를 Feature 모델 - OWL - Class 모델변환 알고리즘을 이용하여 XMI 문서를 생성한다.

이렇게 생성된 feature 모델의 XMI 문서는 StarUML을 통해 Class 모델로 변환된다. 그림 10은 실제 구현 툴을 이용하여 feature 모델을 XMI Document로 변환시킨 화면을 캡처한 화면이다.

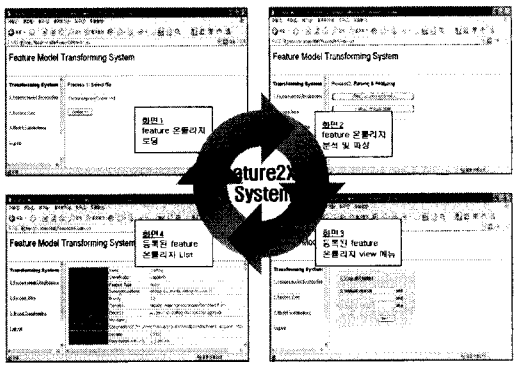

그림 10. Feature 모델 변환 시스템 화면 캡처
Fig. 10 Screen capture of Feature model transformation System

그림 10에서 화면 1은 feature 모델을 온톨로지로 명세한 파일을 로딩하는 화면이고, 화면 2는 로딩된 온톨로지를 분석하고, 파싱하는 화면이다. 파싱된 온톨로지는 DB에 항목별로 저장된다. 화면 3은 저장된 온톨로지를 보여주는 메뉴화면이다. 전체 등록된 feature를 보는 메뉴와 수동 검색하는 기능을 가진다. 화면 4는 등록된 feature list 이다.

VI. 사례적용

논 논문에서 제시한 Feature 모델-OWL-Class 모델간 변환 기법의 사례 연구로 제품 계열 상의 전자 결재 시스템을 대상으로 적용 하였다. 모델 변환은 3장의 그림 1에서 모델 차원의 변환 단계 MT_1~MT_4를 따른다. 즉, 전자결재시스템 도메인을 분석하여 온톨로지, XMI 문서, class 모델로 변환을 한다.

MT_1 단계에서는 전자결재시스템을 feature 모델로 구성한다. feature 메타모델링을 위해 4계층으로 feature를 분류하고, feature를 분석한다. 본 실험에서는 제약 사항으로 feature 메타모델의 4계층 중 능력(Capability) 계층에서 서비스 측면만을 고려하였다. 개별 feature간의 관계성 분석 및 개별 feature 속성 목록 명세는 그림 11에서와 같이 System Service 측면에서의 feature들을 산출하였고, 그 feature들 간의 관계성을 분석하여 트리 Taxonomy 형태로 feature 패턴을 분석한다. 이후 개별 feature들의 속성 목록을 명세한다. 그림 11에서 개별 feature인 'InteriorApproval'의 속성 명세를 보여주고 있다.

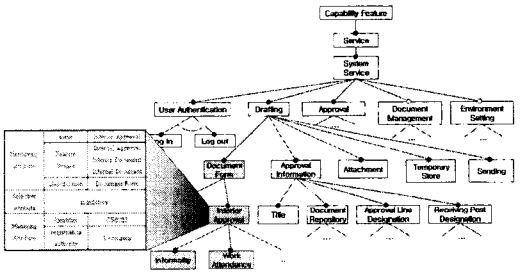

그림 11. 전자결재시스템의 Feature 모델
Fig. 11 Feature Model of Electronic Approval System

다음은 MT_2 단계로, 개별 feature들의 속성을 Feature to OWL 변환 프로파일을 이용하여 변환한다. 그림 12는 전자결재시스템을 온톨로지로 변환한 것이다. 예로, 그림13에서 전자결재시스템의 Document Form feature와 Interior Approval feature는 subclassOf의 관계를 가지므로 〈rdfs:subClassOf〉 〈owl:class rdf:ID="DocumentForm"/〉 〈/rdfs:subClassOf〉로 표현되고, 전자결재시스템의 feature attribute name은 〈nameValue〉 InteriorApproval 〈/nameValue〉로 표현된다.

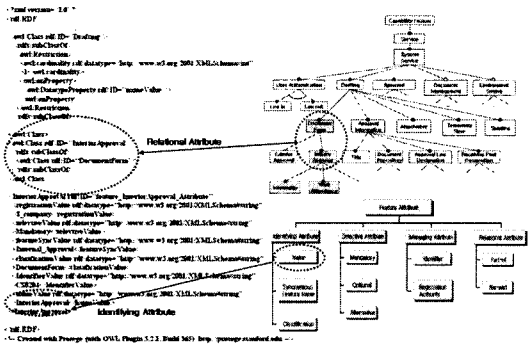

그림 12. 전자결재시스템의 OWL 명세
Fig. 12 Specification of Electronic Approval System using OWL

MT_3 단계에서는 MT_2 단계에서 생성된 OWL 파일을 Feature-to-XMI툴을 이용하여 분석, 저장, 그리고 변환을 한다. 그림 13은 전자결재시스템 온톨로지를 XMI로 변환한 것을 보여준다.

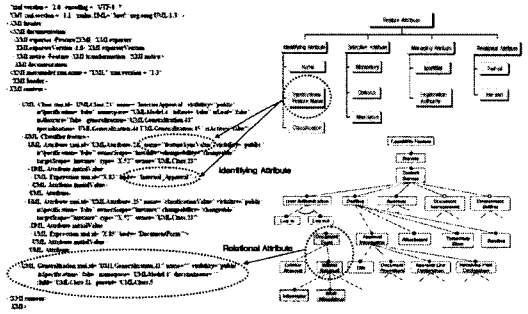

그림 13. 전자결재시스템의 XMI 명세
Fig. 13 Specification of Electronic Approval System using XMI

그림 13에서, 전자결제시스템의 Interior Approval feature는 유사 feature 값으로 Internal_Approval을 갖고, Document Form feature와 Interior Approval간에는 Generalization의 관계를 가진다.

마지막으로 MT_4 단계에서는, MT_3 단계에서 생성된 XMI 문서를 UML 툴(StarUML 6.0)을 이용하여 class 모델로 변환한다. 그림 14는 XMI 문서로 변환된 전자결재시스템을 class 모델로 변환한 것을 나타낸다.

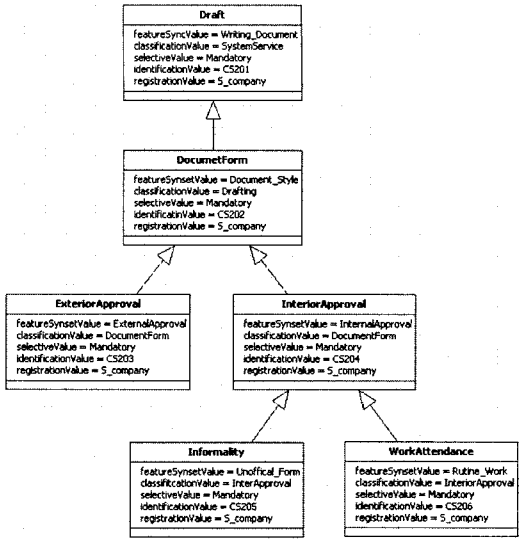

그림 14. 전자결재시스템의 Class 모델
Fig. 14 Class model of Electronic Approval System

전자결제시스템의 각 요소들의 관계와 세부속성들을 class 모델을 통해 모델간 일관성 있게 변환됨을 확인 할 수 있다. 즉, 3장의 그림 1에서 제시된 모델 차원의 적용 기법을 전자결재시스템에 실제 적용하여, feature 모델의 class 모델로의 변환과정과 변환 프로파일과 알고리즘이 실제 모델에 얼마나 잘 적용 되는가에 대해 알 수 있다.

Ⅶ. 평가

6장에서는 온톨로지를 이용한 feature 모델에서 class 모델로 변환 기법의 효율성과 신뢰성을 입증하기 위해 전자결재시스템을 이용한 사례연구를 하였고, 이장에서는 Hassn Gomaa의 방법[7]과 Xin Liu의 방법[8]과 본 논문에서 제시한 방법을 비교 평가 한다. 제시 방법에 많은 재사용 공헌 사례를 적용해서 그 실효성을 입증해야 하나, 현실적으로 다수의 프로젝트 적용은 어려운 실정이다. 그래서 비 정량적인 비교방법을 사용해 변환의 방법 및 지원 툴의 지원여부 그리고 품질측정 비교 매트릭스로 재사용성, 일관성 및 확장성을 가지고 기존연구와 분석하였다. 평가 항목은 모델간 비교를 위한 요소로 자체 선정하였으며, 변환 profile, 변환 툴 제공, 메타모델 차원 변환, 확장성, 재사용성, 일관성, 그리고 변환 방법 등 7가지 요소이다. 그리고 각 항목별 1점의 점수를 부여하고, 총점은 7점으로 설정한다. 이때, 각 비교 매트릭스의 점수 1은 백분율을 기준으로 적용하였다. 모델변환 방법을 평

가하기 위한 평가 산정 기준은 표 8과 같다.

표 8. 평가 산정 기준
Table. 8 Evaluation standard

구분		평가 항목	점수
변환 방법	변환 profile	모델간 변환 profile을 제공하는가?	
		• feature to class 모델 변환 프로파일 제공	1
		- feature to ontology 변환 프로파일 제공	(0.5)
		- class to ontology 변환 프로파일 제공	(0.5)
	Meta model 차원 변환	모델 변환시 속성기반으로 하였는가?	
		• feature, class 모델 속성 맵핑	0.5
		• 온톨로지 속성 맵핑	0.5
변환 툴 지원	변환 툴	모델 변환 툴을 제공하는가?	
		• feature to class 변환 툴 제공	1
		- feature to ontology 변환 툴 제공	(0.5)
		- ontology to class 변환 툴 제공	(0.5)
	변환 자동화	모델 변환은 자동화 되었는가?	
		• feature to class 자동 변환	1
		- feature to class 반자동 변환	(0.5)
품질	재사용성	특정 도메인에서 사용된 방법을 다른 도메인에서 다시 사용 가능한가?	
		• 수정 없이 재사용 가능	1
		- 일부 속성 수정하여 재사용 가능	(0.5)
	일관성	모델변환에 있어서 일정한 체계를 제공하는가?	
		• 정형화된 변환프로세스 제공	1
		- 휴리스틱한 방법 제공	(0.5)
	확장성	feature 모델의 속성이 추가되거나, 새로운 속성이 발견되더라도 각 모델에 적용 가능한가?	
		• feature 모델의 속성이 추가 되었을 경우	1
		- 일부 변경 후 가능	(0.5)
계			7

표 8에서 '•'은 주요 평가항목으로, 모델변환 방법이 해당 항목에 해당사항 없을시 하부 항목 '-'에서 한 개를 선택 한다.

평가 등급은 G(Good), M(Medium), P(Poor) 3단계로 나누고, 표 9에서와 같이 정의한다. 충족도는 평가항목을 만족하는 정도로 해당 항목에서 모델이 획득한 평가 값을 해당 항목 총 평가 값으로 나눈 값이다.

표 9. 평가 등급 산정 기준
Table. 9 Evaluation grade standard

평가 등급	충족도	정 의
G(Good)	67% 이상	평가 항목의 대부분을 충족함
M(Medium)	66%~34%	평가 항목을 충족하지만, 일부 수정 필요
P(Poor)	33% 이하	평가 항목 대부분을 충족하지 않음

표 10은 표 8의 모델 평가 산정 기준과 표 9의 평가등급 산정기준을 이용하여 기존 변환 모델들과 본 논문에서 제시한 모델을 비교한 것이다.

표 10. 기존 방법과 제시방법간의 비교 평가
Table. 10 Comparison evaluation between existing method and proposal method

구분		계	Hasan Gomaa		Xin Liu		Proposal method	
			점수	평가등급	점수	평가등급	점수	평가등급
계		7	4	M	3	M	6	G
변환 방법	변환 profile	1	1	G	0.5	M	1	G
	Meta model 차원변환	1	0.5	M	0	P	1	G
변환 툴 지원	변환 툴	1	0	P	1	G	1	G
	변환 자동화	1	0	P	0.5	M	0.5	M
품질	재사용성	1	1	G	0	P	1	G
	일관성	1	0.5	M	0.5	M	0.5	M
	확장성	1	1	G	0.5	M	1	G

표 8의 평가 항목을 이용하여 측정한 결과 Hassan Gomaa는 4(M), Xin Liu는 3(M)의 값을 갖고, 본 논문에서 제안한 기법은 6(G)의 측정값을 가진다. 결과적으로 본 기법이 다른 기법들보다 평가항목에 대한 충족도가 높다.

그림 15는 표 10에서 측정한 충족도를 바탕으로 모델들 간의 변환 기법을 비교 평가한 방사형 그래프이다.

-■-는 제안된 기법, -▲-는 Xin Xiu의 기법, 그리고 -●-: Hasan Gomaa의 기법의 항목별 충족도를 나타내고 있다. 그래프에서 알 수 있듯이, 본 논문에서 제시한 기법이 다른 두 기법보다 변환 profile, 확장성, 재사용성 등 여러 면에서 우수하고, 평가 항목을 만족하는 총 충족도 측면에서도

우수함을 알 수 있다.

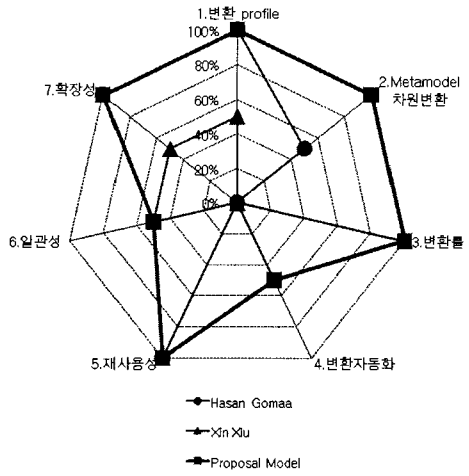

그림 15. 모델변환 방법 비교
Fig. 15 Model transformation methods comparison

결국, 제시하는 변환방법을 사용해서 특정 도메인에 대한 정보구조모델 즉, feature 기반 개발방법에 의해 산출된 feature 모델을 상이한 객체기반 개발방법에서 요구되는 class 모델로 제시하는 변환 방법 및 자동화 툴을 사용함으로써 기 구축된 feature 모델을 재사용하여, 새로이 class 모델을 작성하지 않음으로써 재사용 모델의 품질을 보장 할 수 있고, 개발 기간을 단축 할 수 있을 것이다.

Ⅷ. 결론

제품계열공학에서 재사용을 위해 상이한 모델링 개발방법들간에 모델 변환에 대한 연구가 이루어졌지만, 이 연구들은 변환되는 모델링 요소가 충분치 못하며, 메타모델 차원에서 변환을 다루고 있지 않아 모델의 확장 변경에 유연한 접근이 어려웠다. 본 논문에서는 메타모델 상에서 온톨로지를 사용한 feature 모델과 class 모델간의 일원화된 변환 방법을 제시하였다. 이를 위해, feature 모델-ontology-class 모델에 대한 메타모델을 재정립하고, 각 메타모델별 모델링 요소에 대한 속성을 정의한다. 이 속성들에 기반하여 feature 모델과 ontology간 그리고 ontology와 class 모델간의 변환 규칙 프로파일을 정하고 이를 명확히 표현하기 위해 집합 이론과 명제논리로 정의한다. 이러한 변환의 자동화를 위해 변환 알고리즘을 생성하고, 지원 툴을 구현하였다.

또한 제시한 변환 규칙, 과정 및 툴을 사용해 전자결재시스템 모델 변환을 통해 그 실효성을 보였다.

기대효과로는 메타모델 상에서 온톨로지를 사용한 feature 모델을 class 모델로 일원화된 변환 방법을 제시함으로써, 기 구축된 feature 모델을 class 모델로 변환하여 상이한 개발방법에 재사용할 수 있다는 것이다. 또한, 온톨로지의 OWL을 사용하여 웹상에서 두 모델간 변환을 제공한다. 특히, 온톨로지를 사용한 의미적 변환의 모호성을 해소시킬 수 있으며, 기존 툴과 개발 툴을 사용한 모델간 변환의 자동화로 모델의 일관성을 유지시켜준다.

향후 연구로는 본 연구에서 다루지 않은 class 모델의 속성 중 feature 모델의 속성과 상이한 속성에 대한 맵핑방법과 UML의 다른 모델들의 메타속성을 정의하여 feature 모델과의 변환 방법에 대한 연구가 필요하다.

참고문헌

[1] Clements, P. and Northrop, L, Software Product Lines: Practices and Patterns, Addison-Wesley, Boston, 2001.

[2] Kang, K. C., Kim, S., Lee, J., et al, "FORM: A Feature-Oriented Reuse Method With Domain Specific Reference Architecture", Annals of Software Engineering, Vol. 5, pp. 143-168, 1998.

[3] Kang, K., Lee, J., Donohoe, P, "Feature-Oriented Product Line Engineering", IEEE Software, Vol. 19, No. 4, pp.58-65, 2002.

[4] Lee, K., Kang, K., Lee, J, "Concepts and Guidelines of Feature Modeling for Product Line Software Engineering", Software Reuse: Method, Techniques, and Tools, Springer Berlin/Heidelberg, Vol. 2319, pp.62-77, 2002.

[5] Don Batory, "Feature models, Grammars, and Propositional Formulas", Software Product lines, Springer Berlin/Heidelberg, Vol. 3714, pp.7-20, 2005.

[6] Kang, K, C., Kim, S., Lee, J., et al, "Feature-Oriented Engineering of PBX software for Adaptability and Reusability", Software Practice and Experience, Vol. 29, No. 10, pp.875-896, 1999.

[7] H. Gomaa, Designing Software Product Lines with UML, Addison-Wesley, 2005.

[8] Xin Liu, Generating UML Diagrams using Feature Diagrams for Software Product Line,

pp. 1- 72, 2006. http://alexandria.tue.nl/extra1/afstversl/wsk-i/xinliu2006.pdf

[9] Xin Peng, Wenyun Zhao, Yunjiao Xue, Yijian Wu, "Ontology-based Feature Modeling and Application-Oriented Tailoring", Reuse of Off-the-Shelf Components, Springer Berlin/Heidelberg, Vol. 4039, pp.87-100, 2006.

[10] 이윤수, 김태석, 양진혁, 정인정, "소프트웨어 공학적 방법을 이용한 온톨로지의 효율적인 설계 및 생성에 관한 연구", 한국 정보처리학회 추계학술 발표대회, Vol. 11, No.2 , pp.645-648, 2004.

[11] Stefan Wendler, Mapping XMI / UML to DAML+OIL, Automatische University, 2002.

[12] 이순복, 김진우, 송치양, 김영갑, 권주흠, 이태웅, 김현석, 백두권, "소프트웨어 제품계열공학의 Ontology 기반 회처 공통성 및 가변석 분석 기법", 정보과학회논문지, Vol. 34, pp.196-211, 2007.

[13] 이지홍, 양진혁, 송종수, 정인정, "UML을 이용한 효율적인 온톨로지 재사용에 관한 연구", 한국지능정보 시스템학회 2006 춘계 학술대회논문집, pp. 265-269, 2006.

[14] C.Y. Song and D.K. Baik, "A Layered Metamodel for Hierarchical Modeling UML", International Journal of Software Engineering and Knowledge Engineering, Vol. 13, No. 2, pp. 191-214, 2003.

[15] Michael K. Smith, Chris Welty, Deborah L. McGuinness, OWL Web Ontology Guide, W3C Recommendation 10, 2004.

저자소개

김 동 리

2001년 육군사관학교 전자공학과 학사
2007년~현재 고려대학교 컴퓨터·전파통신공학과 석사과정
관심분야 : 소프트웨어 공학, Ontology, 모델링&시뮬레이션

송 치 양

1985년 한남대학교 전산학과 학사
1987년 중앙대학교 전산학과 석사
2003년 고려대학교 컴퓨터학과 박사
1990년~2005년 한국통신 중앙연구소 책임연구원
2005년~2007 상주대학교 소프트웨어 공학과 조교수
2007년~현재 경북대학교 소프트웨어 공학과 조교수
관심분야 : UML 모델링 기술, 소프트웨어 개발방법, IP-TV 서비스

강 동 수

1997년 해군사관학교 전기공학 학사
2006년 국방대학교 전산학 석사
2008년~현재 고려대학교 컴퓨터·전파통신공학과 박사과정
관심분야 : 소프트웨어 공학, SOA, System Engineering

백 두 권

1974년 고려대학교 수학과 학사
1977년 고려대학교 대학원 산업공학과 석사
1983년 Wayne State Univ. 전산학과 석사
1985년 Wayne State Univ. 전산학과 박사
2002년 고려대학교 정보통신대학 학장
1986년~현재 고려대학교 컴퓨터학과 교수
1989년~현재 한국 정보처리학회 부회장
1992년~현재 ISO/IEC JTC1/SC32 국내위원회 위원장
1999년~현재 정보통신진흥협회 데이터 기술위원회 의장
2005년~현재 한국 시뮬레이션학회 고문
관심분야 : 데이터베이스, 소프트웨어 공학, 데이터 공학, 컴포넌트 기반 시스템, 메타데이터 레지스트리, 정보 통합, 프로젝트 매니지먼트