Product-Line에서의 Feature Model의 명세화 방안

송재승[0], 김민성, 박수용
서강대학교 소프트웨어 공학 연구실
(anthony[0], mskim)@selab.sogang.ac.kr
sypark@ccs.sogang.ac.kr

Feature Model Specification in Product Line

Jae-Seung Song[0], Min-Sung Kim, Sooyong Park
Dept. of Computer Science & Engineering, Sogang University

요 약

빠르게 변화하는 시장의 요구에 대응하고자 특정 도메인에 속하는 애플리케이션 간의 재사용을 높이려는 Product-Line에 대한 연구가 진행되고 있다. Product-Line에서는 도메인 내의 여러 애플리케이션 간의 차이점과 공통점을 분류하는데 Feature Modeling이라는 개념을 주로 사용하고 있다.
이러한 Feature를 추상화 하여 메타 모델로 나타내고 정형화 기법을 통하여 명세화 한다면, 기존의 Feature 모델에서 발견해내지 못하는 feature들과 관련 요소들을 추출하고, 명세화를 통한 통일된 해석이 가능하며, Feature Model에 대한 reasoning, 충돌에 대한 예측 및 협상 등의 지원이 가능할 것이다.
따라서 본 논문에서는 Multi-paradigm 명세 언어를 제공하여주는 KAOS 방법을 적용하여, 추상화 수준에서의 메타 모델을 제안하고, Feature를 명세화 하고 습득하는 방안을 제시하고자 한다.

1. 서 론

특정 도메인 내에서 개발하고자 하는 여러 애플리케이션 간의 공통점과 차이점을 발견하여 재사용을 높이려는 " Product Line" 에서는 주로" Feature Model" 을 사용하여 이러한 분석을 한다.

FODA(Feature Oriented Domain Analysis)[1]에서는 feature 분석을 통하여 도메인 내에서의 애플리케이션들의 공통점 및 차이점들을 분석을 해낸다. 이러한 분석단계는 Product-Line에서 중요한 단계로서 많은 연구[2]가 이루어지고 있지만, feature를 Formal 또는 Semi-Formal한 형식으로 명세화 하는 방법에 대한 연구는 부족한 실정이다.

목표 지향 요구공학 분야에서는 KAOS[3]라는 방법을 사용하여 추상화 레벨에서 목표를 모델링하고 명세함으로써, 목표에 대한 reasoning과 증명 및 검증 그리고 충돌에 대한 관리를 가능하게 하였다.

본 논문에서는 Multi-paradigm 명세 언어를 제공해주는 KAOS 방법을 사용하여 Feature를 정형화 시키는 방안을 제시하고, Feature와 관련 메타 모델의 요소들을 습득하는 프로세스를 제안한다.

2장에서는 Product Line에서 사용되어지는 Feature Modeling기법과 정형화 기법에 대해서 관련연구로서 기술하며, 3장에서는 먼저 Feature에 대한 메타 모델을 제시하고 이를 KAOS 방법론에서 제공하는 Multi-paradigm 명세 언어를 사용하여 명세화 하는 방안을 기술한다. 또한 Feature와 메타 모델에서 제시되어진 각 요소들을 습득하는 프로세스를 제시하고 각 단계에 대한 기술을 한다. 마지막으로 4장에서는 결론 및 향후 연구과제를 기술한다.

2. 관련연구

2.1 FODA in Product Line

Product Line에서는 Feature를 사용하여 애플리케이션 간의 공통점과 차이점을 구분해 내는데 이 과정에서 FODA를 사용한다. Feature-oriented Domain Analysis(FODA)[1]에서 제공하는 feature oriented 개념은 도메인 내에서의 공통점과 차이점들을 발견해 내는데 초점을 맞추고 있기 때문이다. 여기서 의미하는 Feature는 " 사용자 입장에서 보는 소프트웨어 시스템 또는 시스템의 주요하거나 명백한 모양, 품질 또는 특징" 으로 이해되어진다.

FODA에서는 Feature들은 서로 alternative, optional 그리고 mandatory 타입의 특별한 link로써 연결되어지며, 이러한 관계는 AND/OR 그래프를 통하여 나타내어진다. [1]

이러한 Feature는 사용자와 개발자간의 의사소통을 원활하게 해주며, 여러 애플리케이션 간의 공통점과 차이점을 발견해 낼 수 있다는 장점이 있는 반면에, 새로운 시스템 또는 분석이 덜 되어진 시스템에 적용하는 데에는 부적합하며, 정형화에 대한 연구가 부족하여 이에 대한 연구가 이루어 지고 있다. [4]

2.2 정형화 명세 기법

정형화 명세 기법은 시스템이 가져야 할 속성들의 집합들을 정형화된 언어와 추상화 된 레벨로써 표현하는 것이다. 이러한 정형화 명세기법은 현재 소프트웨어와 주변 환경이 이미 확립되어지고, 객체와 오퍼레이션들이 정확히 만들어지고 난 명세 프로세스의 후반부에서 점차적으로 많이 쓰여지고 있으며, 대부분의 언어들은 이러한 하위 레벨의 명세에 초점을 맞추고 있다. [5]

정형화 명세 기법은 명세 paradigm에 따라 [표 1]과 같이 구분되어진다.

정형화 명세 기법	지원 언어
History-based	ERAE, TRIO
State-based	Z, VDM
Transition-based	STATECHART, SCR
Algebraic	LARCH, ASL
Operational	PAISLEY, GIST

[표 1] 정형화 명세 종류와 지원 언어

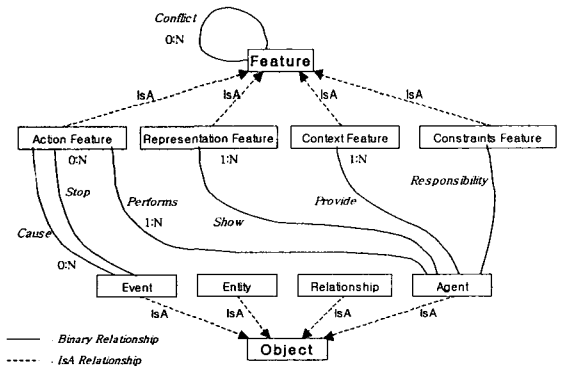
하지만, 이러한 명세 언어들의 제한된 사용범위, 하위 레벨에서의 명세 등의 문제점으로 인하여, KAOS[3] 같은 Multi-paradigm 명세 언어 및 추상화를 지원하는 언어에 대한 연구가 이루어지고 있다.

3. Feature 명세화 기법

본 장에서는 Feature를 메타 모델로 추상화 하여 보다 상위 레벨의 관계를 도출해 내고, 이를 정형화 하여 명세화 하는 방법과 메타 모델에서의 요소들을 습득하는 프로세스에 대해 알아본다.

3.1 Feature 메타 모델

Feature를 정형화 하여 표현하기 위해서는, 먼저 그 의미가 수식으로 표현 가능한 레벨까지 refinement가 되어 있어야 하며, 그 구조가 명확히 정의 되어 있어야 한다.



[그림 1] 개념적 Feature 메타 모델

위의 [그림 1]은 본 논문에서 제안하는 Feature의 개념적인 메타 모델을 나타낸다. Feature는 그 특성에 따라서 Action, Representation, Context 그리고 Constraint Feature로 구분되어지며, Objects는 Event, Agent, Relationship 그리고 Entity로 구분되어진다.

이러한 메타 모델을 통해서 앞으로 사용될 정형화 언어의 구조를 결정할 수 있으며, 추상화 레벨을 높임으로써 도메인에 독립적인 Meta-concept을 만들 수 있는 장점이 있다.

3.2 Feature 명세

FODA에서 Feature는 시스템 또는 소프트웨어 시스템의 주요한 특징, 성능, 상태, 품질 등 기능적, 비기능적인 요소 뿐 아니라 모든 특징들을 포함하고 있기 때문에 이를 앞서 제시한 메타 모델에 따라 구조화 하여야 하며, multi-paradigm 언어를 써서 표현해야 한다.

Feature 메타 모델의 각 요소들은 정형화 되어 나타내어지며, 특히 Feature는 다음과 같은 기본적인 구조를 가지게 된다.

```
Feature [Name]
   Type - Type of feature
   Commonality - Commonality of feature
   InstanceOf - Category of the Feature
   Concerns - Object related to feature
   BindingTime - when bind to the system
   InformalDef - Describe feature informally
   FormalDef - Specify feature formally
   Priority - To manage conflict
```

[그림 2] Feature의 기본 명세

이러한 기본적인 명세 이외에 Feature의 범주에 따라서, Action feature일 경우에는 Input, Output 그리고 *Pre*, *Trigger*, *Post* Condition에 대한 명세가 추가되어지며, Constraint feature는 EnsuredBy, UnderResponsibilityOf 등의 명세가 추가된다. .

```
Feature [WakeUpAlarm]
   Type capability
   Commonality optional
   InstanceOf operation
   Concerns User, Alarm, Schedule
   BindingTime reuse time
   InformalDef When your PCS is powered off…
   FormalDef (∀pcs, schedule:sh)
      pcs.Power=off∧pcs.sh=true∧
current=sh.time
      ⇒ pcs.Alarm=on
  Priority 0
```

[그림 3] 정형화된 Feature의 기본명세

위의 [그림 3]은 정형화 명세 기법에 맞추어 Feature를 명세화 한 기본명세를 보여주고 있다. 정형화 명세에 사용되어지는 기호와 수식은 Z[6]에서 사용되어지는 것들을 사용했다

3.3 Feature-Oriented 습득 프로세스

메타 모델에서의 각 요소들은 습득 프로세스에 의해서 습득되어질 수 있다.

다음 [그림4]은 본 논문에서 제시하는 습득 프로세스의 각 단계를 나타내고 있다.

1) 시스템의 Feature와 관련 Objects들을 찾아낸다.
2) Agents들을 찾아내고, 각 Agents들의 Feature들과의 관계를 설정한다.
3) 하위 수준의 Feature들로부터 Constraints를 찾아낸다.
4) Objects와 Action Feature를 추가하고, 이로부터 파생되는 Elements들을 추가한다.
5) 각 Feature들을 강화한다.

[그림 4] Feature-Oriented Acquisition Process

각 단계는 메타 모델의 요소들을 습득하기 위한 과정으로써, What, How 그리고 Why를 통해서 설명되어진다. 위에서 제시되어진 각 단계들은 순차적으로 정렬이 되어 있지만, 때에 따라서는 (예를 들면, 첫번째와 두 번째 Step의 경우) 중첩되어 실행되어질 수 있으며, 모든 단계는 순환적으로 반복되어지고, 각 단계에서의 변경 사항들은 다른 단계에 전파되어져야 한다.

각 단계를 통해서 Feature Analyst는 메타 모델 그래프의 노드들을 이동하고, 각 요소들을 습득할 수 있으며, 정형화를 통하여 새로운 요소들과 기존의 것들을 추가하고, 강화 할 수 있다.

3.4 기대효과

본 논문에서 제시하는 방법을 통해 Product-Line에서 사용되어지는 Feature를 정형화 시킴으로써, 다음과 같은 것들을 기대하여 볼 수 있다.

첫째, Feature를 정형화 시키고, 이를 통해 Feature를 Reasoning함으로써, 비정형적으로 표현된 Feature에서 간과되어질 수 있는 여러 가지 Feature들을(특히, 비기능적인 Feature나 Constraint관련 Feature) 추가로 얻어낼 수 있다.

둘째, 기존의 통합된 Feature Model에서 Feature를 Action, Representation, Context 그리고 Constraint로 분류하여 나타냄으로써, 복잡도를 줄 일 수 있다.

셋째, Feature에 대한 정형화를 통해서, 동일한 해석을 제공하여 줄 수 있다.

넷째, Feature를 구조화 함으로써, Feature들간의 충돌 발생시 이를 Feature의 properties를 통해서 협상을 함으로써 해결할 수 있다.

이 외에도 여러 가지들을 Feature의 정형화를 통해서 기대할 수 있을 것이다.

4. 결론 및 향후 연구 과제

소프트웨어가 점점 크고 복잡해지며, 사용자의 요구가 급변하는 시장의 요구에 대응하고자 소프트웨어의 재사용에 대한 관심이 높아지고 있다. Product-Line은 특정 도메인 내에서의 재사용을 극대화 하려는 개발 방법으로써 최근 SEI를 중심으로 활발히 연구되어지고 있다.

본 논문에서는 Product-Line에서 같은 도메인 내의 여러 애플리케이션 간의 공통점과 차이점을 도출해내는 방법으로 사용되는 Feature에 대한 정형화된 명세 방법을 제시함으로써 Feature에 대한 reasoning, 검증, 충돌 관리 등을 가능하게 하였다. 이 과정에서 KAOS방법론을 사용하여 추상화 레벨에서의 Feature의 개념적인 메타 모델을 도출해 내었으며, KAOS에서 제공되어지는 Multi-paradigm 언어를 사용하여, Feature와 여러 메타 모델을 정형화하여 명세화 하는 방법을 제시하였다.

향후에는 이러한 연구를 좀 더 보안하여, Product-Line에서 사용되어지는 비즈니스 전략적인 측면이 추가되어 확장된 명세화 기법에 대한 연구가 필요하다.

5. 참고 문헌

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

[2] C. Kang, S. Kim, J. J. Lee, K. Kim, E. Shin, M. Huh, FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures, *Annals of Software Engineering*, 5:143-168, (1998).

[3] A. Dardenne, A. van Lamsweerde and S. Fickas, " Goal-Directed Requirements Acquisition" , *Science of Computer Programming*, Vol.20, 1993, 3-50.

[4] Kyo C. Kang, Sajoong Kim, Jaejoon Lee, Kwanwoo Lee, "Feature-Oriented Engineering of PBX Software for Adaptability and Reusability," Software Practice & Experience, Vol. 29 No.10, pp. 875-896, 1999.

[5] A. van Lamsweerde, " Formal Specification: a Roadmap" . *In The Future of Software Engineering*, A. Finkelstein(ed.), ACM Press, 2000.

[6] B. Potter, J. Sinclair and D. Till, *An Introduction to Formal Specification and Z*. Second edition, Prentice Hall, 1996.