외부 유스케이스를 이용한 확장 모델링 기법

조준수*, 정기원**
*SK C&C 개발담당
**숭실대학교 컴퓨터학부
e-mail: bcto@naver.com

An Extended Modeling Technique Using External Use Cases

Junsoo Cho*, Kiwon Chong**
*Dept. of Development, SK C&C
**School of Computing, Soongsil University

요 약

시스템 개발을 위해서는 시스템이 제공하는 기능을 명료하게 기술함은 물론 목표 시스템이 참조하는 외부 시스템의 기능을 명확하게 식별하여야 한다. 이는 목표 시스템의 범위를 명확하게 한정하기 위해 매우 중요하다. 그러나 현재의 유스케이스 모델링 기법에서는 외부 시스템은 액터로만 표현된다. 이는 외부 레거시 시스템을 간결하게 표현할 수 있다는 장점이 있으나, 외부 시스템의 기능 중 목표 시스템이 참조하는 유스케이스만을 식별하기 어렵게 만드는 단점도 갖는다. 이러한 불명확성은 유스케이스 명세 시 모델 작성을 어렵게 하여 유스케이스 모델의 이해도를 저하시키는 원인이 된다. 본 논문에서는 외부 유스케이스를 이용한 확장 모델링 기법을 제시한다. 확장 모델링은 레거시 시스템에 존재하는 외부 유스케이스를 표현하고, 내부 유스케이스와 관계성을 가질 수 있도록 확장 메커니즘을 지원한다. 확장을 위해서는 UML 확장 메커니즘 중 하나인 스테레오타입(Stereotype)을 활용하며, 따라서 기존 유스케이스 모델과의 호환성을 그대로 유지함으로써, 기존 모델링 기법과 일관되게 적용 가능하다.

1. 서론

시스템 개발을 위해서는 시스템이 제공하는 기능을 명료하게 기술하는 것과 더불어 목표 시스템이 참조하는 외부 시스템의 기능을 명확하게 식별하여야 한다. 이는 목표 시스템의 범위를 명확하게 한정하기 위해 매우 중요하다.

유스케이스(Use Case)는 시스템의 기능을 표현하기 위한 수단으로 널리 사용되고 있으며, 사용자 요구사항을 매우 효과적으로 표현할 수 있다[2][3][4]. 유스케이스가 적용되는 시스템 경계는 서브젝트(Subject)로 정의되며, 시스템이나 서브시스템, 패키지, 클래스 등이 될 수 있다. 서브젝트 외부에 존재하면서 목표 시스템과 상호작용하는 대상은 액터(Actor)로 정의된다[10].

그러나 현재의 유스케이스 모델링 기법에서는 외부 시스템은 액터로만 표현된다. 이는 외부 레거시 시스템을 간결하게 표현할 수 있다는 장점이 있으나, 외부 시스템의 기능 중 목표 시스템이 참조하는 유스케이스만을 식별하기 어렵게 만드는 단점도 갖는다. 이러한 불명확성은 유스케이스 명세 시 모델 작성을 어렵게 하여 유스케이스 모델의 이해도를 저하시키는 원인이 된다[1][5][6].

본 논문에서는 외부 유스케이스를 이용한 확장 모델링 기법을 제시한다. 확장 모델링은 레거시 시스템에 존재하는 외부 유스케이스를 표현하고, 내부 유스케이스와 관계성을 가질 수 있도록 확장 메커니즘을 지원한다. 확장을 위해서는 UML 확장 메커니즘 중 하나인 스테레오타입(Stereotype)을 활용한다.

2. 관련 연구

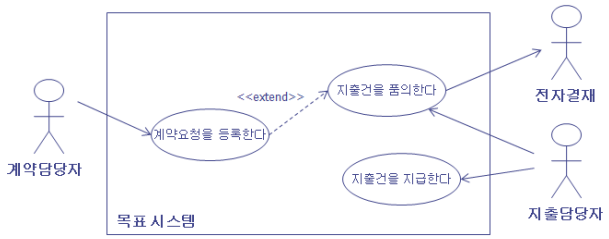
유스케이스 모델의 확장을 위하여 Regnell은 메타모델을 추가하는 방식을 적용하였다. Regnell은 유스케이스의 개념적 분할인 에피소드(Episode)라는 개념을 제안하였는데, 유스케이스는 다수의 에피소드들로 구성되며, 각각의 에피소드는 이벤트를 가질 수 있다. 그러나 그는 에피소드와 액티비티(Activity) 간 개념적 차이를 제시하지 않았고, 유스케이스와 에피소드, 이벤트 간의 계층적 분할에 집중하였다[8]. Yu 및 Rui 등은 Regnell의 연구를 기반으로 유스케이스 재구조화를 위한 기법을 추가로 제안하기도 하였으나, 이들의 연구는 유스케이스 모델링의 개선보다는 리팩토링(Refactoring) 기법 자체에 주안점을 두고 있다[9][11].

반면 타 모델링 기법을 접목시킨 연구로는 Nakatani를 들 수 있다. 그들은 유스케이스 자체가 Composite 패턴 구조를 가질 것을 제안하였다. 그러나 현재 유스케이스와 액티비티간의 개념 차이가 불명확하고, 유스케이스간 실행 순서를 정의할 수 없다고 볼 때, 유스케이스의 Composite 패턴 구조화는 유스케이스 모델의 복잡도만을 증가시키는 결과를 초래할 수도 있다[7].

메타모델의 추가나 타 모델링 기법과의 접목은 유스케이스 모델링 기법의 개선을 통해 모델의 표현력을 증가시킬 수 있다. 그러나 액터와 관련된 기존 유스케이스 모델의 근본적인 모호성을 개선시키지 못하며, 나아가 메타모델의 추가는 기존 유스케이스 모델과의 호환성 문제를 야기할 수 있다.

3. 기존 유스케이스 모델링의 문제점

유스케이스는 시스템의 기능을 표현하기 위한 수단으로 널리 사용되고 있으며, 유스케이스 모델은 시스템의 전체 맥락을 이해하는데 매우 유용한 도구이다. 유스케이스를 활용하여 목표 시스템의 기능을 명세하고자 할 때, 시스템 내부에 존재하는 유스케이스가 외부 시스템의 기능을 참조하는 경우, 액터로 정의된 레거시 시스템과 관계성을 갖게 된다. 그림 1에서 "지출건을 품의한다" 유스케이스는 "전자결재" 레거시 시스템과 관계를 갖는다.



(그림 1) 유스케이스 모델 (예시)

전자결재 시스템은 다양한 기능을 가질 수 있다. 그러나 그림 1에서 목표 시스템은 전자결재 시스템이 제공하는 기능 중에서 구체적으로 어떤 기능을 참조하는지 표현하지 못하며, 따라서 목표 시스템이 참조하는 외부 시스템의 범위를 한정하지 못하게 된다.

이를 해결하기 위해서는 목표 시스템이 참조하는 레거시 시스템의 기능, 즉 외부 유스케이스를 식별하여 표현할 수 있어야 한다. 또한 목표 시스템 내부에 존재하는 내부 유스케이스가 외부 유스케이스를 참조할 수 있도록 이들 간의 관계성을 설정할 수 있어야 한다. 그리고 이러한 확장 모델링은 기존 유스케이스 모델링 기법과 일관되게 적용 가능하여야 한다.

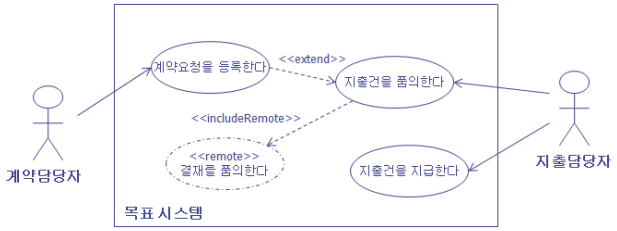
4. 유스케이스 확장 모델링 기법

유스케이스 확장을 위해서는 UML 스테레오타입(Stereotype) 확장 메커니즘을 활용한다. 스테레오타입은 기본 메타모델을 재분류하고, 새로운 속성을 추가할 수 있는 확장 메커니즘으로서, 모델링 요소의 추가적인 분류기준을 제공한다.

외부 유스케이스는 외부 시스템에 존재하는 유스케이스로서, 목표 시스템 내부에 존재하는 내부 유스케이스와 구별된다. 외부 유스케이스를 표현하기 위하여 기존 유스케이스를 확장한 <<remote>> 스테레오타입(Stereotype)을 정의한다.

<<remote>>로 정의된 유스케이스는 외부 유스케이스를 목표 시스템 내에서 명시적으로 표현하기 위해 사용된다. <<remote>> 유스케이스는 의미상으로 액터의 개념을 포함하며, 일점쇄선 윤곽선을 갖는 타원으로 표현된다. <<remote>> 유스케이스는 외부 유스케이스와 1:1의 관계를 가지며, 유스케이스 명세를 갖지 않는다. 이는 외부 유스케이스의 기능수행 절차까지 표현할 필요는 없기 때문이다.

외부 유스케이스를 참조하는 모든 내부 유스케이스는 <<remote>> 유스케이스와 관계를 가질 수 있다. 내부 유스케이스와 <<remote>> 유스케이스 간 관계 설정을 위해 기존의 <<include>> 및 <<extend>> 관계를 확장한 <<includeRemote>> 및 <<extendRemote>> 스테레오타입을 정의한다. <<includeRemote>>는 <<remote>> 유스케이스를 포함(include)시킬 때 적용하며, <<extendRemote>>는 <<remote>> 유스케이스를 확장(extend)시킬 때 적용한다.

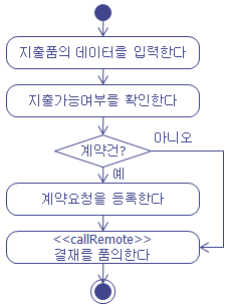


(그림 2) <<remote>> 유스케이스를 적용한 모델

그림 2는 그림 1에 <<remote>> 유스케이스를 적용하여 재구성한 것이다. <<remote>> 유스케이스로 정의된 "결재를 품의한다" 유스케이스는 전자결재 외부 시스템에 정의된 유스케이스를 나타내며, 기존의 해당 액터와의 관계가 아닌, 유스케이스 간 관계로 표현된다. 따라서 "지출건을 품의한다" 유스케이스와 <<includeRemote>> 관계를 통해 연관된다. 이를 통해 기존 액터와의 관계만으로는 식별이 어려웠던 외부 시스템의 기능이 명시적으로 식별 가능하다.

<<remote>> 유스케이스는 내부 유스케이스와 관계를 가질 수 있으며, 타 <<remote>> 유스케이스 또는 액터와의 관계는 허용되지 않는다. 이는 유스케이스를 모델링 할 때 목표 서브젝트 외부에 존재하는 액터 간 관계를 표현하지 않는 것과 동일하다.

외부 유스케이스를 활용하는 경우, 유스케이스의 보다 명확한 명세가 가능하다. 그림 3은 액티비티 다이어그램(Activity Diagram)을 이용하여 "지출건을 품의한다" 유스케이스를 명세한 예이다.



(그림 3) 액티비티 다이어그램을 이용한 유스케이스 명세

외부 유스케이스는 외부 시스템에 정의된 기능이며, 이에 대한 참조는 외부 시스템 기능을 호출함을 의미한다. 외부 유스케이스를 호출하는 액션을 표현하기 위하여 <<callRemote>> 스테레오타입을 정의한다.

그림 3 액티비티 다이어그램에서 다른 액션들은 목표 시스템 내부에 정의된 기능인 반면, "결재를 품의한다" 액션은 전자결재 레거시 시스템에 정의된 기능이며, 따라서 <<callRemote>> 액션으로 정의된다. <<callRemote>> 액션은 외부 기능을 호출하며, 자동화 도구가 사용되는 경우, 외부 유스케이스로부터 쉽게 추출이 가능하다.

텍스트 기반의 유스케이스 명세에서도 외부 유스케이스를 활용할 수 있다. 그림 4의 유스케이스 명세에서 "결재를 품의한다" 액션 앞의 "(callRemote)" 접두사를 통해, 해당 액션이 외부 유스케이스의 호출임을 알 수 있다. 그러나 텍스트 기반의 유스케이스 명세는 자동화 도구의 장점을 활용하는 데 한계가 있으며, 따라서 그 활용의 폭이 제한적이다.

유스케이스명 : 지출건을 품의한다
기본흐름: 1. 지출품의 데이터를 입력한다. 2. 지출가능여부를 확인한다. 3. 계약건의 경우, 계약요청을 등록한다. 4. (callRemote) 결재를 품의한다. 예외흐름 2.1 지출가능액 부족 시 오류메시지를 출력하고, 종료한다.

(그림 4) "지출건을 품의한다" 유스케이스 명세

유스케이스 확장 모델링을 위한 UML 스테레오타입은 표 1과 같다. 유스케이스 모델의 확장은 UML 메타모델을 추가하는 방식을 지양하고, UML 스테레오타입 확장 메커니즘을 이용한다. 따라서 기존 유스케이스 모델과의 호환성을 그대로 유지함으로써, 기존 모델링 기법과 일관되게 적용 가능하다.

<표 1> 유스케이스 확장 모델링을 위한 스테레오타입

확장요소 및 표기법	UML 모델	설명
외부 유스케이스 <<remote>>	유스케이스	외부 서브젝트에 정의된 유스케이스를 참조
유스케이스간 관계 <<includeRemote>>	관계	외부 유스케이스와 내부 유스케이스 간 포함 관계
유스케이스간 관계 <<extendRemote>>	관계	외부 유스케이스와 내부 유스케이스 간 확장 관계
외부기능 호출 액션 <<callRemote>>	액션	외부 유스케이스를 호출하는 액션

5. 결론

목표 시스템의 기능을 명세하기 위한 수단으로는 유스케이스가 널리 활용되고 있다. 그러나 현재의 유스케이스 모델링에서 기존 레거시 시스템은 액터로만 표현 가능하다. 이는 목표 시스템이 참조하는 외부 시스템 기능을 명확하게 식별하지 못함으로써, 시스템의 범위를 한정하지 못하게 한다. 또한 유스케이스 모델 작성을 어렵게 하고, 유스케이스 모델의 이해도를 저하시키는 원인이 된다.

본 논문에서 제안하는 유스케이스 확장 모델링은 외부 시스템에 정의된 외부 유스케이스를 목표 시스템 내에서 명시적으로 표현하도록 지원한다. 외부 유스케이스는 레거시 시스템에 정의된 기능을 표현하며, 목표 시스템의 범위를 명확히 정의하는데 활용된다. 외부 유스케이스는 확장된 관계를 통해 내부 유스케이스와 연관된다.

유스케이스 모델의 확장은 UML 메타모델을 추가하는 방식을 지양하고, UML 스테레오타입 확장 메커니즘을 이용한다. 따라서 기존 유스케이스 모델과의 호환성을 그대로 유지함으로써, 기존 모델링 기법과 일관되게 적용 가능하다.

참고문헌

[1] Armour, Phillip G., "The Laws of Software Process", CACM, Jan 2001

[2] Cockburn, Alistair, "Writing Effective Use Cases", Addison-Wesley, 2000

[3] Collins-Cope, Mark, "The Requirements/Service/Interface(RSI) Approach to Use Case Analysis", Technology of Object-Oriented Languages and Systems, Aug 1999

[4] Firesmith, Donald G., "Use Case Modeling Guideline", Technology of Object-Oriented Languages and Systems, Aug 1999

[5] Isoda, Sadahiro, "On UML2.0's Abandonment of the Actors-Call-Use- Cases Conjecture", Journal of Object Technology, Vol. 4 No. 6, August 2005

[6] Lilly, Susan, "Use Case Pitfalls: Top 10 Problems from Real Projects Using Use Cases", Proceedings of the Technology of Object-Oriented Languages and Systems, 1999

[7] Nakatani, Takako, Urai, Tetsuya, Ohmura, Sou, and Tamai, Tetsuo, "A Requirements Description Metamodel for Use Cases", 8th Asia-Pacific Software Engineering Conference, 2001

[8] Regnell, Bjorn, "Requirements Engineering with Use Cases", Doctoral Thesis, 1999

[9] Rui, Kexing and Butler, Greg, "Refactoring Use Case Models: The Metamodel", The 26th Australasian conference on Computer science, 2003

[10] Rumbaugh, James, Jacobson, Ivar, and Booch, Grady, "The Unified Modeling Language: Reference Manual", 2nd Ed., 2005

[11] Yu, Wei, Li, Jun, and Butler, Greg, "Refactoring Use Case Models on Episodes", The 19th International Conference on Automated Software Engineering, Sep 2004