서비스 지향 컴퓨팅을 위한 확장 유스케이스 모델링

Extended Use Case Modeling for Service-Oriented Computing

조 준 수* 정 기 원**
Junsoo Cho Kiwon Chong

요 약

공급자 시스템이 제공하는 기능의 조합으로 새로운 서비스를 정의하기 위해서는 서비스 조합에 포함되는 기능, 즉 유스케이스가 명확히 식별되어야 한다. 그러나 현재의 유스케이스 모델링은 명세하고자 하는 목표 서브젝트의 외부 시스템을 액터로만 표현함으로써, 서비스 구성에 참여하는 외부 유스케이스를 명확하게 표현하지 못한다. 이러한 불명확성은 유스케이스 모델의 이해도를 저하시키는 원인이 되며, 목표 시스템이 참조하는 외부 시스템의 범위를 명확하게 한정하지 못하게 한다. 본 논문에서는 서비스 지향 컴퓨팅을 위한 확장 유스케이스 모델링 기법을 제시한다. 확장 모델링은 외부 서브젝트에 정의된 외부 유스케이스 및 컴포넌트를 목표 서브젝트 내에서 명시적으로 표현하도록 지원한다. 외부 유스케이스는 공급자 시스템에 정의된 기능을 표현하며, 목표 시스템이 참조하는 범위를 명확히 정의하는데 활용된다. 유스케이스 구현을 위한 외부 컴포넌트와의 관계성은 유스케이스 실체화를 통해 이루어진다. 유스케이스 모델의 확장은 UML 스테레오타입 확장 메커니즘을 이용한다. 따라서 기존 유스케이스 모델과의 호환성을 그대로 유지함으로써, 기존 유스케이스 모델링 기법과 일관되게 적용 가능하다.

ABSTRACT

It is necessary to identify the use cases of the supplier systems in order to define new service using their functions. Current use case modeling techniques, however, do not represent the external use cases clearly, because the external system is represented only by an actor. This ambiguity of the modeling technique declines the comprehensiveness of the model, and does not limit the scope of the target system explicitly. In this paper, we propose the extended use case modeling technique for service-oriented computing. This modeling technique enables the definition of external use cases and components. They are used to limit the boundary of the target system, and the relationship between them are modeled by the realization of external use cases. The proposed technique uses UML stereotype extension mechanism. so the compatibility with current use case modeling technique is preserved.

☞ KeyWords : Use case, Service-orientation, Modeling, Stereotype, Profile, 유스케이스, 서비스 지향, 모델링, 스테레오타입, 프로파일

1. 서 론

서비스 지향 컴퓨팅(Service-Oriented Computing)은 서비스를 단위로 시스템을 구성하는 컴퓨팅 패러다임으로서, 공급자 시스템(Supplier System)이 제공하는 기능을 조합하여 서비스 계층을 구성하고, 이를 통하여 사용자가 원하는 기능, 즉 서비스를 제공한다. 사용자 비즈니스 프로세스는 비즈니스 목적을 달성하기 위한 서비스의 집합으로 구성되며, 비즈니스 규칙과 제약사항 등을 적용하여 서비스들을 조직화한다. 실제 기능은 공급자 시스템에 의해 제공된다.[9][12]

UML 유스케이스(Use Case)는 시스템의 기능을 표현하기 위한 수단으로 널리 사용되고 있으며, 사용자 요구사항을 매우 효과적으로 표현할 수 있다.[3][4][6] 유스케이스가 적용되는 시스템 경계는 서브젝트(Subject)로 정의되며, 시스템이나 서브시스템, 패키지, 클래스 등이 될 수 있다. 서

* 정 회 원 : 숭실대학교 대학원 컴퓨터학과 재학 중
(박사 수료) bcto@naver.com
** 정 회 원 : 숭실대학교 컴퓨터학부 교수
chong@ssu.ac.kr
[2009/02/17 투고 - 2009/02/18 심사 - 2009/03/26 심사완료]

브젝트 외부에 존재하면서 목표 시스템과 상호작용하는 대상은 액터(Actor)로 정의된다.[15]

서비스 지향 컴퓨팅에서 서비스를 구성하기 위해서는 공급자 시스템에 정의된 기능들 중 서비스 구성에 필요한 기능을 식별하고, 이를 조합하여 서비스를 구성할 수 있어야 한다. 따라서 공급자 시스템의 기능을 명세하고 있는 유스케이스를 추출, 목표 시스템의 유스케이스 모형 작성에 활용할 수 있어야 한다. 그러나 현재 유스케이스 모델링 기법에서는 서브젝트 경계 외부에 존재하는 대상은 액터로만 표현된다. 이는 외부 시스템을 간결하게 표현할 수 있다는 장점이 있으나, 외부 시스템 기능 중 서비스 구성에 필요한 유스케이스만을 식별하기 어렵게 만드는 단점도 갖는다.[8][10]

이러한 문제점은 목표 시스템 내부에 존재하는 내부 유스케이스를 조합하는 서비스에 대해서도 나타날 수 있다. 유스케이스는 서브젝트를 단위로 표현되는데, 동일한 시스템 내에 둘 이상의 서브젝트가 존재하는 경우, 이들 각각은 상대 서브젝트를 액터로 표현하게 되며, 이 경우 위에서 기술한 동일한 문제점을 보이게 된다.

서비스를 정의하기 위해서는 서비스 조합에 포함되는 기능, 즉 유스케이스가 명확히 식별되어야 한다. 그러나 현재의 유스케이스 모델링은 명세하고자 하는 목표 서브젝트의 외부에 존재하는 서브젝트 또는 레거시 시스템을 액터로만 표현함으로써, 서비스 구성에 참여하는 유스케이스를 명확하게 표현하지 못한다. 이러한 불명확성은 유스케이스 명세 시 모델의 이해도를 저하시키고, 목표 시스템이 참조하는 범위를 명확하게 한정시키지 못함은 물론, 자동화 도구의 이점을 충분히 활용하지 못하게 하는 제약으로 나타난다.

본 논문에서는 서비스 지향 컴퓨팅을 위한 확장 유스케이스 모델링 기법을 제시한다. 확장 모델링은 서비스를 명세함에 있어서 목표 서브젝트 외부에 존재하는 외부 유스케이스 및 외부 컴포넌트를 정의할 수 있도록 한다. 또한 유스케이스 실체화를 통해 이들간의 관계성을 정의함으로써, 기존 유스케이스 모델링 기법과 일관되게 적용 가능하도록 지원한다. 확장을 위해서는 UML 확장 메커니즘인 스테레오타입(Stereotype)을 활용한다.

2장에서는 서비스 지향 컴퓨팅 및 UML 프로파일의 기본 개념과 유스케이스 확장에 관한 선행 연구를 소개하고, 3장에서는 유스케이스를 이용한 서비스 모델링의 문제점을 지적한다. 4장에서는 서비스 지향 컴퓨팅을 위한 유스케이스 확장 모델을 제시하며, 5장에서는 연구 사례를 제시하고, 6장에서 결론을 내린다.

2. 관련 연구

2.1 서비스 지향 컴퓨팅

서비스 지향 컴퓨팅은 서비스를 단위로 시스템을 구성하는 컴퓨팅 패러다임으로서, 공급자 시스템이 제공하는 기능을 조합하여 서비스 계층을 구성하고, 이를 통하여 사용자가 원하는 기능, 즉 서비스를 제공한다. 서비스는 상태를 갖지 않는 기능적 컴포넌트로서, 하위 계층이 제공하는 기능을 조합하여 제공하고자 하는 기능을 구성, 전달하는 역할을 수행하며, 실제 기능은 공급자 시스템에 의해 제공된다.[9][12]

서비스 지향 컴퓨팅은 추상화 수준에서 볼 때, 크게 오퍼레이션(Operation), 서비스(Service), 비즈니스 프로세스(Business Process) 등으로 구분할 수 있다. 오퍼레이션은 단일 트랜잭션을 의미하며, 서비스는 오퍼레이션의 논리적인 집합을 의미한다. 비즈니스 프로세스는 비즈니스 목적을 달성하기 위한 서비스의 집합으로 구성되며, 비즈니스 규칙과 제약사항 등을 적용하여 서비스들을 조직화한다. 그러나 서비스의 명세에 대해서는 합의된 형태가 존재하지 않는다.[19]

2.2 유스케이스의 확장

지금까지 유스케이스의 확장은 주로 메타모델(Meta-model)을 추가 적용하거나 타 모델링 기법을 접목시키는 등 기존 유스케이스 모델링 기법을 개선시키는 데 집중되어 왔다.

메타모델의 적용을 통한 확장으로는 Regnell이 제시한 유스케이스 메타모델이 있다. Regnell은 유스케이스의 개념적 분할인 에피소드(Episode)라는 개념을 제안하였다. 유스케이스는 다수의 에피소드들로 구성되며, 각각의 에피소드는 이벤트를 가질 수 있다. 그러나 그는 에피소드와 액티비티 간 개념적 차이를 제시하지 않았고, 유스케이스와 에피소드, 이벤트 간의 계층적 분할에 집중하였다.[13] Yu 및 Rui, 조준수 등은 Regnell의 연구를 기반으로 유스케이스 재구조화를 위한 기법을 추가로 제안하기도 하였으나, 이들의 연구는 유스케이스 모델링의 개선보다는 리팩토링(Refactoring) 기법 자체에 주안점을 두고 있다.[1][14][18]

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 실행순서를 정의할 수 없다고 볼 때, 유스케이스의 Composite 패턴 구조화는 유스케이스 모델의 복잡도만을 증가시키는 결과를 초래할 수도 있다.[11]

이러한 방식의 확장 유스케이스 모델링은 유스케이스 모델링 기법의 개선을 통해 모델의 표현력을 증가시킬 수 있다. 그러나 서브젝트 단위로 표현되는 기존 유스케이스 모델의 근본적인 모호성을 개선시키지 못한다. 나아가 추가적인 메타모델의 도입은 기존 유스케이스 모델과의 호환성 문제를 야기할 수 있다.

2.3 UML 프로파일

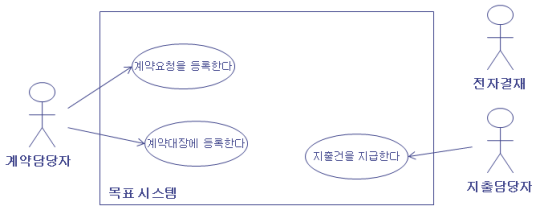
UML은 새로운 메타모델을 추가하지 않고, 기존 메타모델에 새로운 속성을 추가함으로써 UML의 확장이 가능한 메커니즘을 제공한다. 확장 메커니즘에는 스테레오타입(Stereotype)과 꼬리표 값(Tagged Value), 그리고 제약사항(Constraint)의 세 가지 형태가 있으며, 이들의 집합이 UML 프로파일이다.[15]

스테레오타입은 기본 메타모델을 재분류하고, 새로운 속성을 추가할 수 있는 확장 메커니즘으로서, 모델링 요소의 추가적인 분류기준을 제공한다. 모델링 요소의 추가적인 특성은 꼬리표 값을 통해 설정 가능하며, 모델링 요소가 갖는 제약은 제약사항을 통해 표현 가능하다.

현재 OMG에서는 다양한 업무 도메인 및 플랫폼에 적용할 수 있는 프로파일을 표준으로 제정했거나 진행 중에 있으며, 이러한 노력은 학계 및 산업계로까지 점차 확산되고 있다.

3. 유스케이스를 이용한 서비스 모델링

유스케이스(Use Case)는 시스템의 기능을 표현하기 위한 수단으로 널리 사용되고 있다. 유스케이스를 활용하여 서비스를 정의하고자 할 때, 목표 시스템이 하나의 서브젝트로 구성되는 경우, 기존의 유스케이스 모델링 기법을 활용한 목표 시스템 명세는 비교적 용이하다.



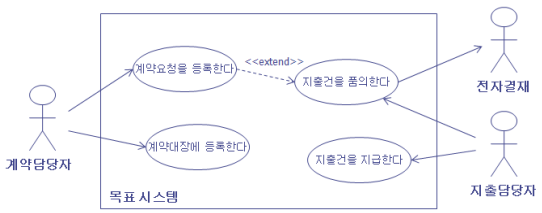
(그림 1) 유스케이스 모델 (예시)

그림 1은 하나의 서브젝트로 구성된 목표 시스템을 유스케이스와 액터, 그리고 이들 간의 관계를 이용하여 명세한 유스케이스 모델을 보이고 있다. 상기 모델에 그림 2의 명세를 갖는 “지출건을 품의한다” 서비스를 추가하기로 한다.

유스케이스명 : 지출건을 품의한다
관련 액터 : 지출담당자, 전자결재
기본흐름: 1. 지출품의 데이터를 입력한다. 2. 지출가능여부를 확인한다. 3. 계약건의 경우, 계약요청을 등록한다. 4. 결재를 품의한다. 대안흐름 2.1 지출가능액 부족 시 오류메시지를 출력하고, 본 유스케이스를 종료한다.

(그림 2) "지출건을 품의한다" 유스케이스 명세

"지출건을 품의한다" 서비스의 기본흐름 중에서 "계약건의 경우, 계약요청을 등록한다"는 목표 시스템 내부 서비스를 활용하며, "결재를 품의한다"는 전자결재 시스템과 연계하여 처리된다. 따라서 서비스가 추가된 유스케이스 모델은 그림 3과 같다. "지출건을 품의한다" 서비스는 기존 서비스의 조합을 위해 "계약요청을 등록한다" 서비스와 <<extend>> 관계를 이용해 확장되며, 전자결재 시스템과 연계된다.

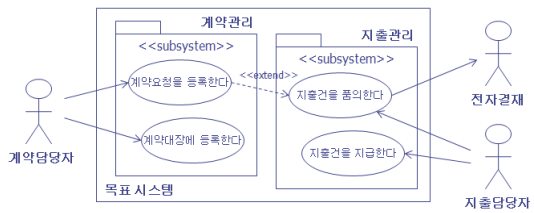


(그림 3) 서비스가 추가된 유스케이스 모델

"계약요청을 등록한다" 기능은 목표 시스템 내부에 존재하며, 따라서 명확한 식별이 가능하다. 반면 전자결재 시스템은 레거시 시스템이며, 목표 시스템에서는 전자결재 시스템이 갖는 기능들 중 "결재를 품의한다" 하나의 기능만을 활용한다. 그러나 이러한 정보는 유스케이스 모델에서는 전혀 나타나지 않고 있다.

유스케이스 모델은 시스템의 전체 맥락을 이해하는데 매우 유용한 도구이며, 목표 시스템의 범위를 명확히 표현할 수 있어야 한다. 따라서 서비스가 공급자 시스템이 제공하는 기능들을 조합하여 제공된다고 할 때, 레거시 시스템이 제공하는 기능의 범위를 명확히 한정시켜야 한다. 그러나 현재의 유스케이스 모델링 기법은 이를 명확히 표현하지 못하며, 이를 해결하기 위해서는 레거시 시스템이 갖는 기능들 중 목표 시스템이 활용하는 기능만을 식별하여 표현할 수 있어야 한다.

이러한 문제점은 목표 시스템이 둘 이상의 서브젝트로 구성되는 경우에도 동일하게 나타날 수 있다. 그림 4는 그림 3의 유스케이스 모델이 계약관리 및 지출관리 서브시스템 단위의 두 개 서브젝트로 분할된 유스케이스 모델을 보이고 있다.

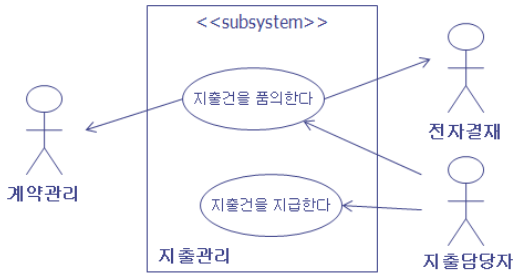


(그림 4) 두 개의 서브젝트를 갖는 유스케이스 모델

그림 4에서 계약관리 서브젝트는 목표 시스템 내부이나, 지출관리 서브젝트를 모델링할 때에는 서브젝트의 정의에 따라 액터로 표현되어야 한다. 따라서 지출관리 서브젝트에 대한 유스케이스 모델은 결과적으로 그림 5와 같이 표현되어야 한다.

그림 5의 유스케이스 모델은 지출관리 서브시스템이 계약관리 서브시스템의 기능을 활용한다는 정보만을 보일 뿐, 구체적으로 어떤 기능을 활용하는가에 대해서는 전혀 표현하지 못하고 있다. 따라서 그림 3에서의 "계약요청을 등록한다" 기능을 활용한다는 정보가 오히려 생략되는 결과를 초래하게 된다.

또한 그림 3에서 표현할 수 있었던 "지출건을 품의한다" 유스케이스와 "계약요청을 등록한다" 유스케이스 간 <<extend>> 관계마저도 표현하지 못할 뿐만 아니라, 관계의 방향성마저 바뀜에 따라 사용자에게 혼란을 초래할 수도 있다.



(그림 5) 지출관리 유스케이스 모델

목표 시스템의 유스케이스 수가 적은 경우, 그림 4와 같이 서브젝트 별로 패키지를 이용하여 분할하고, 하나의 유스케이스 모델로 작성 가능하다. 그러나 패키지를 이용한 분할은 유스케이스들을 동일한 추상화 수준에서 그룹으로 나눌 뿐이며, 목표 시스템의 규모가 커지는 경우 관심의 분할(Separation of Concerns)을 위해서는 서브젝트의 활용이 불가피하다.[2] 그러나 서브젝트 단위로 작성되는 유스케이스 모델은 외부 서브젝트에 속하는 유스케이스와의 관계성 정보가 누락됨으로 인해 모델의 이해도가 저하되는 단점이 있다.

이를 해결하기 위해서는 목표 서브젝트 또는 시스템 외부에 존재하는 유스케이스를 정의하고, 이들과의 관계성 정보를 표현할 수 있어야 한다. 나아가 외부 유스케이스의 실체화 및 관련 외부 컴포넌트까지 정의할 수 있도록 지원함으로써, 확장 유스케이스 모델링을 기존 유스케이스 모델링 기법과 일관되게 적용 가능하여야 한다.

따라서 확장 유스케이스 모델링 기법은 다음을 지원하여야 한다.

1. 외부 서브젝트에 존재하는 외부 유스케이스를 정의할 수 있어야 한다.
2. 외부 유스케이스의 실체화가 가능하여야 한다.
3. 외부 유스케이스 모델링이 기존 모델링 기법과 일관되게 적용 가능하여야 한다.

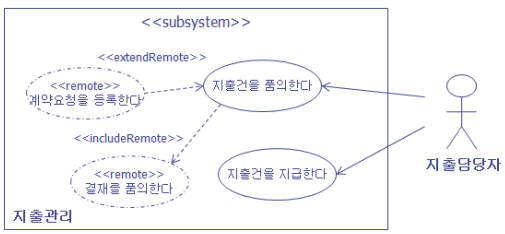
4. 서비스 지향 확장 유스케이스 모델링

4.1 외부 유스케이스

외부 유스케이스는 외부 서브젝트 또는 시스템에 존재하는 유스케이스로서, 목표 시스템 내부에 존재하는 내부 유스케이스와 구별된다. 외부 유스케이스는 외부 서브젝트나 시스템이 제공하는 서비스이며, 목표 시스템은 이들을 조합하여 자신의 서비스를 제공한다. 외부 유스케이스를 표현하기 위하여 기존 유스케이스를 확장한 <<remote>> 스테레오타입(Stereotype)을 정의한다.

<<remote>>로 정의된 유스케이스는 외부 유스케이스를 목표 서브젝트 내에서 명시적으로 표현하기 위해 사용된다. <<remote>> 유스케이스는 의미적으로 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 유스케이스들은 <<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 6) 〈〈remote〉〉 유스케이스를 적용한 모델

그림 6은 그림 5를 <<remote>> 유스케이스를 적용하여 재구성한 것이다. <<remote>> 유스케이스로 정의된 "계약요청을 등록한다"와 "결재를 품의한다"는 각각 계약관리 및 전자결재 외부 서브젝트에 정의된 유스케이스를 나타내며, 기존의 해당 액터와의 관계가 아닌, 유스케이스 간 관계로 표현한다. "계약요청을 등록한다" 유스케이스에서 볼 수 있듯이, 기존 액터와의 관계로 표현할 때 발생하는 정보의 누락이 <<extendRemote>> 관계를 통해 보존됨을 알 수 있다.

<<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.

<<remote>> 유스케이스의 실제 유스케이스는 목표 시스템 내부에 존재할 수도 있고, 외부 레거시 시스템에 존재할 수도 있다. 내부에 존재하는 경우는 목표 시스템이 둘 이상의 서브젝트로 분할되어 있는 경우이며, 해당 유스케이스도 목표 시스템의 일부가 된다. 따라서 자동화 도구가 <<remote>> 스테레오타입을 인지할 수 있다면, 외부 유스케이스를 쉽게 정의할 수 있다. 실제 유스케이스가 레거시 시스템에 존재하는 경우, 목표 시스템에는 해당 유스케이스에 대한 정보가 존재하지 않으며, 따라서 외부 유스케이스는 명시적으로 정의되어야 한다.

외부 유스케이스의 존재 위치의 차이는 해당 유스케이스를 정의하는 방법상의 다름이며, 의미적(Semantic) 또는 문법적(Syntactic) 차이는 존재하지 않는다. 목표 서브젝트 관점에서 이들은 모두 외부에 존재하는 유스케이스이기 때문이다.

4.2 외부 컴포넌트

외부 서브젝트의 컴포넌트는 외부 시스템에 존재하며, 따라서 목표 서브젝트 내에서 직접적으로 참조가 곤란하다. 외부 서브젝트에 존재하는 컴포넌트를 표현하기 위해 <<componentRemote>> 스테레오타입을 정의한다.

<<componentRemote>> 컴포넌트는 외부 시스템에 존재하는 컴포넌트를 표현하며, 일점쇄선 윤곽선을 갖는 컴포넌트 형태로 표시된다. 외부 컴포넌트를 식별하는 이유는 외부 유스케이스와 동일하게 목표 시스템이 참조하는 범위를 한정시키기 위해서이며, <<componentRemote>> 컴포넌트는 이를 목표 서브젝트 내에서 명시적으로 표현하기 위해 사용된다.

그림 5의 "계약관리"와 같이 외부 서브젝트가 목표 시스템 내에 존재하는 경우, 해당 컴포넌트를 통해 외부 컴포넌트를 쉽게 정의할 수 있다. 반면 "전자결재"와 같이 실제 컴포넌트가 레거시 시스템에 존재하는 경우, 해당 컴포넌트를 위한 <<componentRemote>> 컴포넌트를 추가로 정의하여야 한다.

<<componentRemote>>로 정의된 컴포넌트는 개발 단계에서 해당 컴포넌트의 인터페이스로 변환 가능하다. 외부 유스케이스가 목표 시스템 내부에 존재하는 경우, <<componentRemote>> 컴포넌트는 인터페이스로 정의될 수 있으며, 외부에 존재하는 경우 프록시(Proxy) 패턴 등을 적용 가능하다. 자동화 도구를 활용하는 경우, 이러한 변환은 자동으로 수행 가능하다.

4.3 외부 유스케이스의 실체화

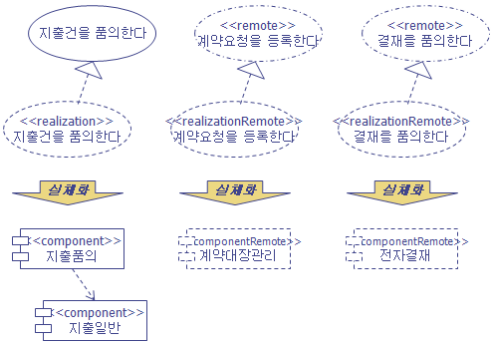
유스케이스는 하나 이상의 유스케이스 실체화(Realization)를 통해 구현된다. 유스케이스는 실체화를 통해 구현하는 컴포넌트(Component)와 맵핑된다. <<remote>> 유스케이스의 실체화를 정의하기 위해 <<realizationRemote>> 스테레오타입을 정의한다.

<<realizationRemote>> 유스케이스 실체화는 <<remote>> 유스케이스와 <<componentRemote>> 컴포넌트 간 실체화 관계를 정의한다. 외부 유스케이스 실체화는 기존 유스케이스 실체화와 동일한 방식으로 표현되며, 이를 통해 내부 유스케이

스를 구현하는 컴포넌트가 접근하는 외부 컴포넌트를 식별할 수 있다.

<<remote>> 유스케이스의 실제 유스케이스가 목표 시스템 내에 존재하는 경우, 외부 유스케이스 실체화는 해당 유스케이스 실체화를 통해 쉽게 정의 가능하다. 반면 실제 유스케이스가 레거시 시스템에 존재하는 경우, 유스케이스 실체화는 추가로 정의되어야 한다.

그림 7에서 "계약요청을 등록한다" 외부 유스케이스 실체화는 실제 유스케이스 실체화를 통해 자동으로 생성 가능하다. 반면 "결재를 품의한다" 유스케이스 실체화는 지출관리 서브젝트에서 추가적으로 정의하여야 한다.



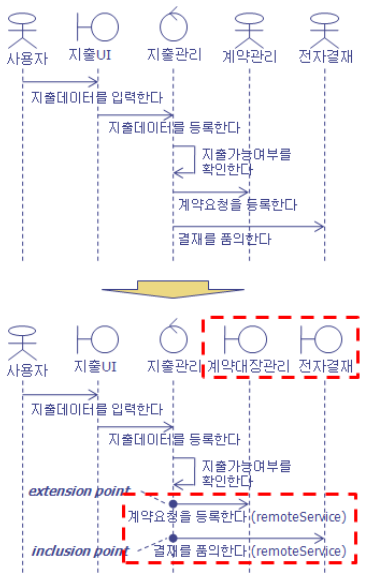
(그림 7) 지출관리 유스케이스의 실체화 (일부)

4.4 유스케이스의 명세

<<componentRemote>> 컴포넌트를 활용하여 유스케이스의 명확한 명세화가 가능하다. 내부 유스케이스 명세에서 외부 유스케이스를 호출하는 <<remoteService>> 스테레오타입을 정의한다.

그림 8의 시퀀스 다이어그램(Sequence Diagram)을 이용한 유스케이스 명세에서 상단은 "계약관리" 서브시스템 및 "전자결재" 레거시 시스템이 액터로만 표현되어 있다. 따라서 해당 기능이 구체적으로 어떤 컴포넌트에 속하는지 알 수 없으며, 참조 범위도 불명확하다.

반면 하단과 같이 <<componentRemote>> 컴포넌트를 적용하게 되면, 보다 명확한 표현이 가능하다. 바운더리 클래스(Boundary class)로 표현된 "계약대장관리" 및 "전자결재"는 식별된 외부 컴포넌트이며, 내부 유스케이스에서 참조하는 범위는 해당 컴포넌트가 갖는 인터페이스로 한정된다. 외부 유스케이스의 호출은 <<remoteService>> 액션을 통해 이루어지며, 이를 명시적으로 기술하기 위해 액션명 뒤에 "(remoteService)"를 병기한다.



(그림 8) "지출건을 품의한다" 유스케이스의 명세

외부 유스케이스의 호출은 해당 유스케이스로의 분기를 의미하며, 따라서 <<remoteService>> 액션은 UML 확장점(Extension Point)을 동반한다. "계약요청을 등록한다"는 <<extendRemote>> 관계에 따른 "extension point"를 가지며, "결재를 품의한다"는 <<includeRemote>> 관계에 따른 "inclusion point"를 갖는다.

서비스 지향 컴퓨팅을 위한 확장 유스케이스 모델링 UML 프로파일은 표 1과 같다. 유스케이스 모델의 확장은 UML 메타모델 계층 중 M2 계층 수준에서 UML 스테레오타입 확장 메커니즘을 적용하였다. 따라서 기존 유스케이스 모델과의 호환성을 그대로 유지함으로써, 기존 모델링 기법과 일관되게 적용 가능하다.

(표 1) 확장 유스케이스 모델링을 위한 UML 프로파일

확장 요소	표기법	UML 모델	설명
외부 유스케이스	<<remote>>	유스케이스	외부 서브젝트에 정의된 유스케이스를 참조
외부 컴포넌트	<<componentRemote>>	컴포넌트	외부 서브젝트에 정의된 컴포넌트를 참조
외부 서비스	<<remoteService>>	액션	외부 서브젝트에 정의된 유스케이스를 호출
유스케이스간 관계	<<includeRemote>>	관계	외부 유스케이스와 내부 유스케이스 간 포함 관계
유스케이스간 관계	<<extendRemote>>	관계	외부 유스케이스와 내부 유스케이스 간 확장 관계
실체화 관계	<<realizationRemote>>	관계	외부 유스케이스의 실체화

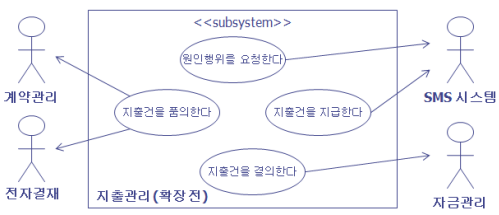
5. 사례 연구 및 평가

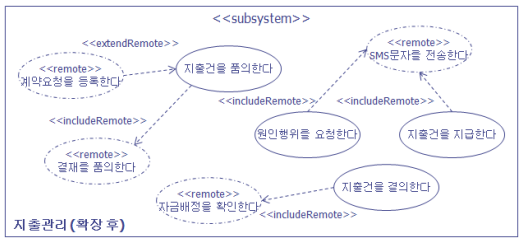
지방재정관리시스템은 지방자치단체가 재원을 확보하고 확보된 재원을 집행하는 일련의 경제활동을 처리하는 시스템으로서, 예산·결산, 수입·지출, 자산·부채의 세 영역으로 구성된다. 본 장에서는 지출관리 영역에 대한 서비스 지향 확장 유스케이스 모델링 사례를 보인다.

5.1 지출관리 유스케이스 모델

지출관리 서브시스템의 확장 전 유스케이스 모델(일부)은 그림 9의 상단과 같다. 유스케이스와 액터는 각각 4개이며, 타 서브젝트 및 외부 레거시 시스템은 액터로 표현되어 있다. 반면 하단 그림은 확장 후 유스케이스 모델을 보이며, 기존 액터에 정의된 기능이 외부 유스케이스로 표현되어 있다. (사용자는 지면의 제약으로 인해 생략한다)

따라서 외부 유스케이스는 해당 액터가 제공하는 서비스가 되며, 지출관리 서브시스템은 해당 외부 서비스를 활용하여 자신의 서비스를 제공하게 된다. 외부 유스케이스는 내부 유스케이스와의 관계성에 따라 <<includeRemote>> 또는 <<extendRemote>> 관계를 갖는다.

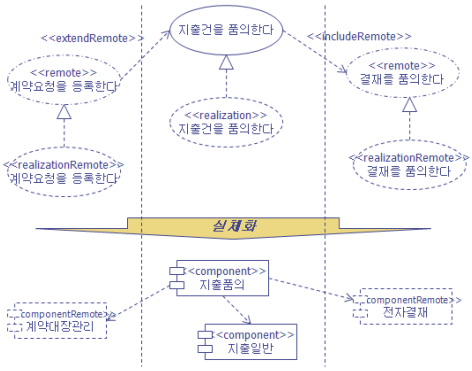




(그림 9) 지출관리 서브시스템의 확장 전·후 모델(일부)

5.2 지출관리 유스케이스의 실체화

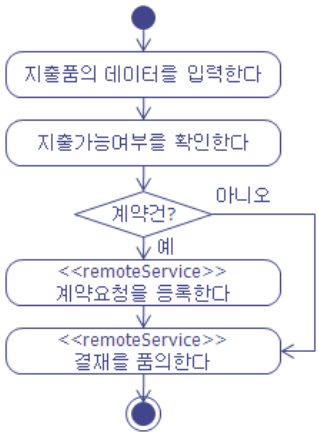
유스케이스 실체화는 유스케이스와 이를 구현하는 컴포넌트 간 관계성을 정의한다. 그림 10의 내·외부 유스케이스의 실체화를 통해 "지출건을 품의한다"는 지출품의 및 지출일반 두 개의 컴포넌트에 의해 구현됨을 알 수 있다. "지출건을 품의한다"는 완결된 서비스를 위해 "계약요청을 등록한다"와 "결재를 품의한다" 서비스를 이용하며, 이들은 각각 계약대장관리 및 전자결재 컴포넌트에 의해 제공됨을 알 수 있다.



(그림 10) 지출관리 유스케이스의 실체화(일부)

5.3 유스케이스의 명세

그림 11은 액티비티 다이어그램(Activity Diagram)을 활용하여 “지출건을 품의한다” 유스케이스를 명세하고 있다. 스테레오타입을 갖지 않는 두 개의 액션은 유스케이스 내부에서 정의한 액션이다. 다른 두 개의 액션은 외부 유스케이스를 호출하며, 따라서 <<remoteService>> 스테레오타입을 갖는다.



(그림 11) “지출건을 품의한다” 유스케이스 명세

5.4 평가

서비스 지향 컴퓨팅 관점에서의 UML의 활용은 크게 서비스의 식별·정의 또는 서비스 통합(Composition) 측면에서 주로 다루어지고 있다. Fatolahi 등은 비즈니스 프로세스 관점에서 유스케이스의 추상화 수준과 그에 대응하는 서비스 수준을 정의하고, 스테레오타입을 활용한 서비스 식별방안을 제시하였다.[5] Heckel 등은 서비스 지향 아키텍처를 위해 UML 클래스 다이어그램 및 협력 다이어그램을 스테레오타입 적용을 통한 확장 방안을 제시하였다.[7]

서비스의 통합을 위한 확장방안으로 Xu 등은 스테레오타입을 적용한 클래스 다이어그램을 제안하고, 액티비티 다이어그램을 통한 서비스 통합 방안을 제시하였다. Skogan 등은 확장된 UML 모델을 활용한 서비스 통합 방법(Method)를 제안하였다.

이들의 연구는 UML 스테레오타입 확장 메커니즘을 활용함으로써 자신이 제공하는 서비스의 명확한 표현이 가능하도록 UML 모델의 표현력을 제고하였다. 그러나 본 논문에서 제안하는 외부 시스템이 제공하는 서비스의 식별이나 이를 구현하는 외부 컴포넌트의 명시적 활용은 지원하지 못한다는 단점이 있다.

6. 결 론

서비스 지향 컴퓨팅은 서비스를 단위로 시스템을 구성하며, 공급자 시스템이 제공하는 기능들을 조합하여 사용자가 원하는 서비스를 제공한다. 목표 시스템의 기능을 명세하기 위한 수단으로는 유스케이스가 널리 활용되고 있다.

유스케이스를 활용하여 서비스를 모델링하는 경우, 레거시 시스템이 액터로만 표현됨으로 인해 목표 시스템이 참조하는 외부 시스템의 범위를 명확하게 한정하지 못하게 된다. 또한 서브젝트 단위로 작성되는 유스케이스 모델링의 특성으로 인해 목표 시스템 내 타 서브젝트도 액터로 표현된다. 이는 단일 서브젝트로 구성된 모델에서 표현 가능한 정보의 누락을 야기시키며, 따라서 유스케이스 모델 작성을 어렵게 하고, 유스케이스 모델의 이해도를 저하시키는 원인이 된다.

본 논문에서 제안하는 확장 유스케이스 모델링은 외부 서브젝트에 정의된 외부 유스케이스 및 컴포넌트를 목표 서브젝트 내에서 명시적으로 표현하도록 지원한다. 외부 유스케이스는 공급자 시스템에 정의된 기능을 표현하며, 목표 시스템이 참조하는 범위를 명확히 정의하는데 활용된다. 유스케이스 구현을 위한 외부 컴포넌트와의 관계성은 유스케이스 실체화를 통해 이루어진다.

유스케이스 모델의 확장은 UML 메타모델을 추가하는 방식을 지양하고, UML 스테레오타입 확장 메커니즘을 이용한다. 따라서 기존 유스케이

스 모델과의 호환성을 그대로 유지함으로써, 기존 모델링 기법과 일관되게 적용 가능하다.

참 고 문 헌

[1] 조준수, 정기원, "서비스 지향 유스케이스 모델링", 2006 한국전자거래학회 종합학술대회, 2006. 11

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

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

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

[5] Fatolahi, Ali and Shams, Fereidoon, "Service Discovery Based on Business Processes: A Practical Approach Using UML", ICSSSM '05, 2005

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

[7] Heckel, Reiko, Lohmann, Marc and Thone, Sebastian, "Towards a UML Profile for Service-Oriented Architectures", Workshop on Model Driven Architecture: Foundations and Applications, 2003

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

[9] Jones, Steve, "Toward an Acceptable Definition of Service", IEEE Software, May/June 2005

[10] 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

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

[12] Papazoglou, M.P. and Georgakopoulos, D., "Service-Oriented Computing", CACM, Oct 2003

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

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

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

[16] Skogan, David, Gronmo, Roy and Solheim, Ida, Web Service Composition in UML

[17] Xu, Yang and Xu, Youwei, "Towards Aspect Oriented Web Service Composition with UML", The 6th ICIS, 2007

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

[19] Zimmermann, Olaf, Krogdahl, Pal and Gee, Clive, "Elements of Service-Oriented Analysis and Design", IBM, June 2004

◐ 저 자 소 개 ◑

조 준 수

1996.02 숭실대학교 전산학과 졸업(학사)
1998.02 숭실대학교 대학원 전산학과 졸업(석사)
1998~현재 숭실대학교 대학원 컴퓨터학과 재학 중(박사 수료)
2006.12～현재 SK C&C 개발담당 과장
관심분야 : 소프트웨어 프로세스 및 품질, 객체지향 시스템 분석/설계, 요구공학, 제품계열 분석 등
E-mail : bcto@naver.com

정 기 원

1967.02 서울대학교 전기공학과 졸업 (공학사).
1981.11 미국 알라바마주립대(헌츠빌) 전산학과 석사.
1983.12 미국 텍사스주립대(알링턴) 전산학과 박사.
1971~1975 한국과학기술연구소 책임연구원.
1975~1990 국방과학연구소 책임연구원.
1990~현재 숭실대학교 컴퓨터학부 교수.
관심분야 : 소프트웨어 공학, 소프트웨어 프로세스, 정보시스템 감리, 전자거래(CALS/EC), 유비쿼터스 컴퓨팅 등
E-mail : chong@ssu.ac.kr