비즈니스 시스템 분석을 위한 유스케이스 중심 개발 방법의 문제점

이혜선○ 박재년
숙명여자대학교
{hslee○, jnpark}@sookmyung.ac.kr

Difficulties of the Use Case Centered Development Methodology for Business System Analysis

HyeSeon Lee○ JaiNyun Park
Sookmyung Womens' University

요 약

비즈니스 시스템의 요구사항 분석을 위해 대부분의 소프트웨어 개발 방법론에서는 UML의 유스케이스 모델링을 이용하고 있으며, 유스케이스는 전체 소프트웨어 개발 프로세스에서 중심적인 역할을 담당하고 있다. 본 논문에서는 유스케이스 중심의 개발 방법을 적용하여 요구사항 명세서로부터 유스케이스를 추출하여 클래스를 도출하기까지의 분석 과정을 살펴보고, 유스케이스 모델링을 수행하는 데 있어서의 문제점에 대해 조사해 봄으로써 유스케이스를 이용한 비즈니스 시스템 분석 과정에서 고려해야 될 사항들을 미리 점검해 볼 수 있도록 하여 초보자도 쉽게 분석할 수 있도록 지침을 제공하고 자 한다.

1. 서 론

비즈니스 업무를 분석하는 데 있어서 주로 사용되고 있는 UML의 유스케이스 모델링은 시스템의 기능을 이해하고 고객과 대화하기 위한 것으로 시스템의 경계, 클래스, 속성, 메소드 그리고 클래스 간의 관계를 식별하는 분석 작업으로, 정확하고 완전한 요구사항 분석은 최종 시스템을 성공적이며 효율적인 비용으로 구축할 수 있는 중요한 요소가 된다[1]. 그러나 요구사항 분석을 위한 유스케이스 모델링 작성에 있어서 분석가의 유스케이스 추출과 시나리오 작성에 대한 추상화 레벨과 관점에 따라 유스케이스 분석의 결과가 서로 다를 수 있으며, 초보자가 분석하는 데 있어서 명확한 지침이 부족하여 많은 시행착오를 겪게 된다. 따라서 유스케이스 분석 모델을 적용하는 데 있어서의 문제점 및 주의할 사항들이 미리 파악된다면 분석을 위한 지침이 될 수 있을 것이다.

본 논문에서는 비즈니스 시스템 분석을 위하여 유스케이스 중심의 개발 방법을 이용한 클래스 도출 과정에 대해 살펴보고, 유스케이스 모델링에서의 문제점에 대해 조사해봄으로써 유스케이스 모델링을 수행하는 데 있어서 고려해야 할 사항들을 미리 점검해 볼 수 있도록 한다.

2. 유스케이스 중심의 개발 방법에서의 클래스 도출 방법

비즈니스 시스템의 요구사항 분석을 위해 주로 사용되는 유스케이스 중심의 개발(UCCD: Use Case Centered Development) 방법은 최종적으로 개발될 시스템을 위한 클래스를 정의하고 정제하기 위한 분석 방법이다[2].

UCCD 개발 과정은 비즈니스 시스템의 요구사항 명세서로부터 유스케이스 다이어그램을 작성하는 유스케이스 모델링과 클래스 다이어그램을 작성하는 객체 모델링을 수행함으로써 분석에 필요한 최종 클래스를 도출한다[2][3].

2.1 유스케이스 모델링(Use Case Modeling)

유스케이스 모델링은 비즈니스 시스템의 요구사항으로부터 액터와 유스케이스를 식별하여 그들 사이의 관계를 표현하는 유스케이스 다이어그램을 작성하고 문서화하는 모델로서 다음의 과정으로 수행된다[2][3][4].

(1) 초기 클래스 식별

UCCD 방법에서의 첫 번째 과정은 요구사항 명세서로부터 주 클래스 목록을 추출하는 것이다. 클래스는 주 클래스(Primary Class)와 지원 클래스(Support Class)로 구분할 수 있는데, 주 클래스는 최종 시스템을 개념화하기 위해 필요한 클래스로 요구사항 명세서로부터 추출되며, 지원 클래스는 초기 시스템을 개념화하기 위해 필요한 클래스가 아닌 분석 단계 이후 또는 설계 단계에서 필요로 하는 클래스이다[2].

요구사항 명세서로부터 주 클래스를 식별하기 위해서는 경험자의 기술이 필요하며, 초보자들이 경험과 기술을 습득하기 위한 지침으로는 Coad-Yourden이 제안한 명사 목록을 클래스 식별을 위한 시작점으로 사용한다. 명

사는 실 세계에서 단순히 객체로 표현할 수 있기 때문에 명사형을 객체 후보로 취하고, 이 명사형이 값을 의미하거나 더 이상 분해될 수 없는 단일 값을 가지는 경우에는 이 명사를 속성으로 바꾼다. 그리고 속성이 같은 객체들을 모아서 클래스로 취하게 된다. 이렇게 추출된 클래스를 주 클래스라 하며 시나리오 작성에 이용된다[2].

(2) 유스케이스 다이어그램 작성

유스케이스 다이어그램은 비즈니스 도메인에서 액터를 식별하고, 비즈니스의 기능인 유스케이스를 식별하여 시스템 내에서 액터와 유스케이스 간의 상호작용을 그림으로 표현한 것이다[5].

액터(Actor)는 시스템을 사용하는 외부 엔티티로 시스템의 정보와 기능을 이용하는 사람, 장치 또는 외부 시스템을 의미하며, 유스케이스(Use Case)는 액터가 시스템에서 어떤 목적을 달성하기 위해 수행하는 일련의 이벤트로 시스템의 기능적인 요구사항을 정의한 것으로 액터에 의해 초기화되며, 시스템은 액터에게 어떤 값을 제공해야 한다[6].

(3) 시나리오 작성

시나리오는 하나의 유스케이스 행위를 설명하기 위한 인스턴스로, 액터의 요청에 대해서 액터와 유스케이스 사이에 어떤 상호작용이 발생하는지 보여주는 이벤트 흐름에 대한 문서화로 표준 템플릿을 사용하여 기술한다[3].

시나리오 기술 방법에는 시스템을 블랙박스로 보고 액터의 행위만을 표현하는 시스템의 외부관점 접근법(External view approach)과 시스템의 내부 행위를 상세하게 표현하는 화이트 박스 형태의 시스템 내부관점 접근법(Internal view approach), 그리고 두 가지 방법을 혼용한 형태가 있다. 시나리오 템플릿은 분석가마다 형태를 달리하여 사용하고 있는 실정이다[4].

2.2 객체 모델링(Object Modeling)

시스템의 행위를 설명하는 유스케이스 다이어그램이 완성되었으면, 시스템의 정적인 구조를 나타내는 객체 모델로 매핑해야 한다. 객체 모델링은 비즈니스의 내부 모델로서, 각각의 유스케이스가 어떻게 실체화되는가를 나타내는 상호작용 다이어그램의 작성을 통하여 객체를 추출하고 정제하여 상세 클래스를 도출하기 위한 방법으로 다음의 과정으로 수행된다.

(1) 유스케이스 실체화를 위한 객체 분류

유스케이스 실체화(Use Case Realization)는 특정 유스케이스가 설계모델 내에서 어떻게 실현될 것인가를 협력하는 객체들의 관점에서 시스템이 갖추어야 할 객체와 관계를 찾아내는 분석 작업으로 상호작용 다이어그램인 콜레보레이션 다이어그램과 시퀀스 다이어그램의 작성을 통하여 수행된다[7].

유스케이스 실체화를 위해 액터의 요구사항을 처리하기 위한 비즈니스 시스템의 내부 행위를 바운드리 객체(Boundary Object), 컨트롤 객체(Control Object), 엔티티 객체(Entity Object)로 식별하여 분류 정의하면서 객체를 추출한다[4][8].

(2) 시퀀스 다이어그램 작성

시퀀스 다이어그램은 사용자, 화면, 시스템 내에서 생성된 객체와 객체 사이의 상호작용과 메시지를 시간적인 순서로 표현한 것으로 유스케이스 시나리오 별로 각각 작성한다[9].

시퀀스 다이어그램은 특정 유스케이스에 대한 이벤트 중심의 플로우로 객체간의 상호작용을 그래픽 형태로 표현해 주기 때문에 객체를 쉽게 식별할 수 있도록 해주며, 시퀀스 다이어그램이 완성되면 필요한 새로운 객체가 도출되는 것이다.

(3) 클래스 다이어그램 작성

클래스는 실행 시 생성되는 객체를 위한 템플릿으로, 같은 특성을 가진 객체들의 집합을 표현한 것으로 개발을 위한 설계 요소이며, 클래스 다이어그램은 실체화 과정을 통하여 추출된 객체들을 분석하여 클래스로 정의하고 클래스간의 관계를 표현한 것이다.

3. 유스케이스 모델링의 문제점 조사

사용자의 요구사항 분석 결과를 유스케이스 다이어그램으로 작성할 때의 유스케이스 함정(Pitfalls)은 다음과 같다[10].

① 시스템 경계가 정의되지 않았거나 변화가 많다.
② 유스케이스를 액터의 관점이 아닌 시스템의 관점에서 작성한다.
③ 액터 명이 일치하지 않는다(Inconsistent).
④ 유스케이스의 수가 너무 많다.
⑤ 액터-유스케이스 관계가 너무 복잡하다.
⑥ 유스케이스 명세서가 너무 길다.
⑦ 유스케이스 명세서가 혼동스럽다.
⑧ 유스케이스 이름(Functional Entitlement)이 유스케이스의 기능을 정확하게 설명하지 못한다.
⑨ 고객은 유스케이스를 이해하지 못한다.
⑩ 유스케이스는 결코 끝나지 않는다.

유스케이스의 함정을 피하기 위해서는 유스케이스 명세서에는 단지 유스케이스 명, 목적, 간략한 설명만을 기술하며, 초기 다이어그램을 검토한 다음에 다이어그램이 정확하고 유스케이스의 명세서가 상세하게 작성되면 유스케이스는 완성된다. 또한 유스케이스 명세서와 함께 최종 다이어그램에 대한 형식적인 검토를 수행함으로써 유스케이스 함정을 해결할 수 있으며, 효율적인 유스케이스 검토를 위해서는 공통적인 유스케이스 문제들을 인식하기 위하여 리뷰 체크리스트를 사용하는 것이 좋다[11].

유스케이스 함정 외에 [12]에서는 유스케이스의 주요 실수 10가지를 다음과 같이 지적하였다.

① 기능적인 요구사항을 작성하지 않는 것
② 속성과 메소드를 설명하지 않는 것

③ 유스케이스를 간단하게 작성하지 않는 것
④ 사용자 인터페이스를 완전하게 무시하지 않는 것
⑤ 바운드리 객체에 대한 명확한 이름 작성을 하지 않는 것
⑥ 사용자의 요구가 아니거나, 수동적으로 작성하지 않는 것
⑦ 시스템의 행위를 무시하지 않는 것
⑧ 대안적인 행위에 대한 명세를 생략하지 않는 것
⑨ 유스케이스 외부의 것에 대해 관심을 두지 않는 것
⑩ <<includes>> 또는 <<extends>>를 사용할 지를 결정하는데 많은 시간을 소비하지 않는 것

또한 [13]에서는 유스케이스를 정의하는데 있어서 10가지 잘못된 가이드라인에 대해 다음과 같이 지적하였다.

① 유스케이스는 요구사항만을 위한 모델인데, 다른 요구사항을 표현하는 것
② 유스케이스 이름에 " 처리", " 실행" 같은 모호한 동사를 사용하는 것
③ 유스케이스의 범위가 모호한 것
④ 유스케이스 명세서에 비 기능적인 요구사항과 UI를 상세하게 포함하는 것
⑤ 초기 유스케이스 다이어그램에서 너무 많은 <<includes>>와 <<extends>>를 사용하는 것
⑥ 비즈니스 규칙을 정의하는데 관계하지 않는 것
⑦ 유스케이스를 생성, 검토, 검증하는 데 있어서 주요 분야 전문가를 포함하지 않는 것
⑧ 유스케이스를 정의하는데 모든 사용자를 포함시키는 것
⑨ 처음부터 초기 유스케이스를 아주 상세하게 작성하는 것
⑩ 작성한 유스케이스가 타당한 지에 대해 검증하지 않는 것

[14]에서는 유스케이스 모델링의 문제점을 다음과 같이 분석하였다.

① 유스케이스의 기능적으로 분해된 명세서로부터 클래스를 설계하기 위한 객체 명세서로의 변환이 어렵다.
② 유스케이스는 클래스에 대해 언급하지 않기 때문에, 개발자들이 각자의 방법으로 객체를 분석하므로 재사용하기가 어렵다.
③ 유스케이스는 비-기능적인 요구사항을 설명하기 어렵다.
④ 유스케이스의 기능적인 테스트는 직관적으로 할 수 있으나, 상세한 구현에 대한 단위 테스트와 비-기능적인 요구사항은 명확하지 않다.

위에서 살펴본 바와 같이 유스케이스 모델링을 작성하는 데 있어서 여러가지 문제점들이 있는 것이 지적되었으며, 대부분의 문제점들은 유스케이스를 식별하고 시나리오를 작성하여 분석하기 위한 구체적이고도 명확한 지침이 부족하기 때문에 분석가마다 유스케이스 분해에 대한 추상화 레벨과 시나리오 작성 수준에 대한 관점이 달라 분석 결과가 다를 수 있다는 것이다. 따라서 유스케이스 모델링에서의 문제점을 미리 잘 파악하여 비즈니스 시스템 분석 시 고려해야 하며, 아울러 문제점을 보완할 수 있는 구체적이고 명확한 지침에 대한 연구도 필요하다.

5. 결 론

본 논문에서는 비즈니스 시스템 분석을 위한 유스케이스 중심 개발 방법의 절차와 문제점에 대해 살펴보았다.

유스케이스 모델링을 수행하는 데 있어서의 여러 가지 문제점 및 고려해야 할 사항들에 대해 미리 점검해봄으로써 시스템 분석의 초보자들이 요구사항 분석을 위한 유스케이스 모델링을 효과적으로 수행할 수 있도록 시스템 분석을 위한 지침으로 활용할 수 있을 것으로 기대된다.

참고문헌

[1]Boeha, B., Egyed, A., " Improving the Life-cycle Process in Software Engineering Education " ,http://sunset.usc.edu/~aegyed/publications/Improving_ the_Life-Cycle_Process_in_Software_Engineering_Education.pdf, 1998.
[2]Evelyn Stiller, Cathie Leblanc, "Project-Based Software Engineering: An Object-Oriented Approach", Addison Wesley, 2002.
[3]Terry Quatrani, " Visual Modeling with Rational Rose and UML" , Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1998
[4]Frank Armour, Granville Miller, "Advanced Use Case Modeling Software Systems", Addison Wesley, 2000.
[5]Francois Coetzee, UML2.0, http://www.xpdian.com/UML2.0.html
[6]Hassan Gomaa, "Designing Concurrent, Distributed, and Real-Time Applications with UML", Addison Wesley, 2000.
[7]Craig Larman, Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (2nd Edition), Prentice Hall, 2001.
[8]Ivar Jacobson, Maria Ericsson, Agneta Jacobson, "The Object Advantage: Business Process Reengineering With Object Technology", Addison Wesley, 1995, pp 113-126.
[9]Francois Coetzee, UML2.0, http://www.xpdian.com/UML2.0.html.
[10]Susan Lilly, " Use Case Pitfalls: Top 10 Problems from Real Projects Using Use Cases" , Proceedings of TOOLS USA '99, IEEE Computer Society, 1999.
[11]Susan Lilly, " How to Avoid Use-Case Pitfalls" , Software Development Magazine, January 2000.
[12]Doug Rosenberg, " Top Ten Use Case Mistakes" , Software Development, February 2001.
[13]Ellen Gottesdiener, " Top Ten Ways Project Teams Misuse Use Cases - and How to Correct Them" , the Rational Edge, June 2002 (Part I), July 2002 (Part II).
[14]"Process and Method: An Introduction to the Rational Unified Process", http://people.cs.uchicago.edu/~matei/CSPP523/lect4.ppt.