비즈니스 시스템을 위한 UML 기반의 분석 모델링의 문제점

이혜선, 박재년
숙명여자대학교 컴퓨터과학과
e-mail : hslee@sookmyung.ac.kr

Difficulties of the UML-based Analysis Modeling for Business System

Hye-Seon Lee, Jai-Nyun Park
Dept. of Computer Science, Sookmyung Women's University

요 약

비즈니스 정보 시스템 개발을 위한 요구사항 분석은 시스템의 결과를 좌우할 수 있는 중요한 과정으로 인식되고 있다. 따라서 대부분의 소프트웨어 개발방법론에서 사용자의 요구사항을 분석하기 위해 주로 사용되고 있는 UML 기반의 Use Case 모델링의 절차와 분석과정에서의 문제점에 대하여 살펴보고, 또한 전반적인 UML 의 문제점을 조사해봄으로써 시스템 개발을 위하여 UML 기반의 모델링을 적용하여 분석할 경우 고려해야 될 사항들을 미리 점검해 볼 수 있도록 지침을 제공하고 자 한다.

1. 서론

지난 20 여년 동안 소프트웨어 프로젝트의 실패율이 높은 것으로 보아 소프트웨어공학 방법론들이 실패였다는 것이 증명되었다[1]. 이는 프로젝트 개발에서 중요한 역할을 담당하는 프로젝트 관리자(PM: Project Manager)가 비즈니스를 잘 이해할 수 없기 때문에 사용자의 빈번한 요구사항 변경이 원인이 되어 프로젝트가 실패하게 된 것이다. 그러므로 사용자의 요구사항을 개발 초기에서부터 얼마나 정확히 분석하느냐에 따라서 프로젝트의 성패가 달려있다고 보여진다. 현재 대부분의 소프트웨어 개발 방법론들은 비즈니스 요구사항 분석 과정에서부터 UML 기반의 모델링을 이용하여 분석하고 있다. 그러나 UML 기반의 모델링은 표기법이 너무 복잡하여 비즈니스 업무를 분석하고 표현하는데 많은 어려움이 존재한다. 따라서 분석 모델링을 적용하는데 있어서의 문제점이 파악된다면 분석을 위한 지침이 될 수 있으므로 요구사항 분석 시 많은 도움을 줄 수 있을 것이다.

본 논문에서는 UML 을 기반으로 하고 있는 CBD 개발 방법론에 대해 살펴보고, 비즈니스 분석을 위한 유스케이스 모델링의 절차와 각 모델링 과정에서의 문제점에 대해 고찰해 보고, 또한 UML 의 전체적인 문제점에 대해 조사함으로써 UML 을 이용한 시스템 분석 시 고려해야 할 사항들을 미리 점검해 볼 수 있도록 한다.

2. UML 기반의 개발 방법론

현재 CBD 방법론을 적용한 다양한 비즈니스용 소프트웨어 프로세스가 개발되어 사용하고 있다. 그러나 CBD 방법론은 컴포넌트를 바라보는 시각과 관점에 따라 차이가 있을 수 있기 때문에 CBD 방법론에 대한 분석과 다양한 프로세스 개발에 대한 연구가 진행되었다[2] [3] [4].

실제 비즈니스 시스템을 개발하기 위해 사용되는 방법론은 자신의 조직 문화나 프로젝트의 성격에 맞게 커스터마이징 하여 사용되고 있으나, 비즈니스 분석 시에는 대부분이 유스케이스 모델링을 적용하여 비즈니스 클래스를 추출하기 위한 분석을 한다. 그러나 모델링 분석과정에서의 구체적인 지침이 없기 때문에 경험자에 많이 의존하고 있는 실정이다.

3. UML 기반 모델링 프로세스

비즈니스 정보 시스템의 전형적인 모델링 분석 방법에서의 모델링 순서는 '비즈니스 모델 → 유스케이스 모델 → 분석 모델 → 설계 모델 → 구현 모델' 단계로 구체화되며, UML 기반의 유스케이스 모델은 시스템의 요구사항을 도출하기 위한 모델로 사용된다.

[표 1]은 객체지향 방법론과 CBD 방법론의 개발 프로세스에서 공통적으로 사용되는 UML 표기법을 추출한 것으로, 유스케이스를 이용한 모델링으로 공통적으로 비즈니스 분석을 하고 있음을 보여주고 있다.

구분	모델링 절차	UML 표기
객체지향 모델링	① 요구사항 분석 ② 비즈니스 모델링 ③ 유스케이스 모델링 ④ 클래스 분석 및 설계 ⑤ 컴포넌트 설계	① Use Case Diagram ② Use Case 명세서 ③ Sequence Diagram 또는 Collaboration Diagram ④ Class Diagram ⑤ Component Diagram
CBD 모델링	① 비즈니스 모델링 ② 유스케이스 모델링 ③ 논리적 모델링 (Business Type Model Interface Specifications Component Specifications Component Object Interface Component Architecture)	① Use Case Diagram ② Class Diagram ③ Class Diagram with <<stereotype>>

[표 1] UML 기반 모델링 프로세스 비교

4. 유스케이스 모델링

요구사항 중심의 개발 프로세스인 유스케이스 모델링은 비즈니스 측면의 비즈니스 프로세스를 정의하는 과정으로, 요구사항 도출/분석작업을 통한 유스케이스 다이어그램 작성, 명세서(시나리오) 작성, 주 클래스 목록 생성, 시퀀스 다이어그램 작성을 통하여 최종적으로 개발 될 시스템을 위한 클래스를 정의하고 정제하기 위한 방법이다. 또한 유스케이스는 개념적인 수준에서의 시스템의 초기행위에 맞추어 모델링 하는 것에서부터 시작하여 점진적이고 반복적인 형태로 요구사항에 맞추어 상세하게 정제된다. [5].

(1) 유스케이스 다이어그램 작성

유스케이스 다이어그램은 시스템과 시스템 외부의 액터 사이의 상호작용을 표현한 것으로, 현재 사용하고 있는 시스템에 대한 고객용 유스케이스를 가지고 새로운 시스템용 유스케이스를 상세하게 작성한다. 또한 논리적이고 순차적인 방법으로 기술함으로써 고객관점의 유스케이스를 추상화시킨 형태로 작성한다.

- 유스케이스 다이어그램 작성 시의 문제점
  - 유스케이스를 추출하는 가이드라인이 애매모호하다. 경우에 따라서 유스케이스가 될 수도 메소드가 될 수도 있다
  - 유스케이스의 범위 결정과 이름도출이 어려워 경험자만 할 수 있다
  - 속성으로 표현해야 할지 유스케이스로 표현해야 할지 구분하기 어렵다.
  - 유스케이스 명세서의 내용은 어느 정도까지의 업무 범위를 표현할 지 결정하기 어렵다.

(2) 유스케이스 명세서 작성

유스케이스 명세서(Descriptions)는 액터의 요청에 대해서 액터와 유스케이스 사이에 어떤 상호작용이 발생하는지 보여주는 사건 흐름을 기술함으로써 완성할 수 있다.

유스케이를 기술하는 방법에는 두 가지 형태가 있다. 시스템을 블랙박스로 보고 액터의 행위만을 표현하는 시스템의 외부관점 접근법(External view approach)과 시스템의 내부행위를 상세하게 표현하는 화이트박스 형태의 시스템 내부관점 접근법(Internal view approach)이 있다[5].

[표 2]와 같은 유스케이스 명세서는 시스템의 외부관점 접근법과 내부관점 접근법을 동시에 사용하여 작성한 형태로, 명세서에서의 주요처리는 예외처리를 제외한 성공적인 처리를 위주로 기술하며, 시스템에서 당연히 리턴되는 내용도 제외하여 표현한다. 이것은 유스케이스 명세서의 복잡성을 줄이고 보다 이해하기 쉬운 형식으로 표현하기 위함이다. 유스케이스 명세서는 분석가마다 다르게 정의하여 작성할 수 있다.

Use Case Name	
Brief Description	
Precondition	
Flow of Events	
Actor View	System View
Postcondition	

[표 2] 유스케이스 명세서 작성 예

(3) 주 클래스 목록 작성

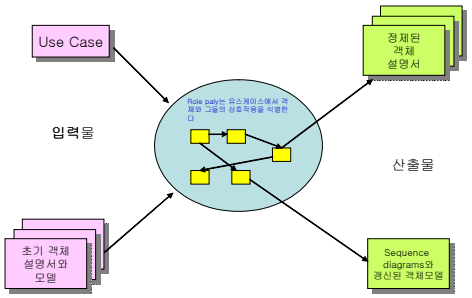
Coad 와 Yourden 은 요구사항 명세서로부터 모든 명사목록을 클래스 식별을 위한 시작점으로 사용하였다. 클래스로 사용하기 위한 조건으로는 정보를 유지하고 있는지, 다른 클래스에 대해 서비스를 제공하는 연산자와 속성이 있는지, 이 명사의 모든 인스턴스는 같은 속성을 공유하는 지 등에 따라 클래스를 구분한다. 이렇게 작성된 클래스는 시퀀스 다이어그램을 통하여 보다 구체적으로 결정된다.

(4) 시퀀스 다이어그램 작성

시스템의 행위를 설명하는 유스케이스 다이어그램이 완성되었으면, 시스템의 정적인 구조를 나타내는 객체 모델로 매핑해야 한다. 유스케이스를 객체모델링으로 통합하는 방법으로는 CRUD(Create, Read, Update, Delete) 메트릭스를 이용하여 객체를 식별하는 방법, CRUD 를 확장하여 각 유스케이스에서의 객체의 구체적인 책임(Responsibility)을 기술하는 UCOM(Use Case Object Model) 메트릭스를 이용하는 방법, 그리고 유스케이스 별로 시퀀스 다이어그램을 작성하여 객체를 추출하는 방법이 있다.

[그림 1]은 유스케이스를 객체로 모델링 하는 방법은 설명한 것이며, 시퀀스 다이어그램의 작성은 브레인스토밍 기법인 Role Playing 을 사용하여 다음과 같은 순서로 작성한다.

① 유스케이스에서 후보객체를 결정한다.
② 후보객체에 대한 구체적인 책임을 식별하고 문서화함으로써 초기 시퀀스 다이어그램을 생성한다. 이 단계에서 새로운 객체가 생성될 수 있다.
③ 새로이 생성된 객체에 대해 객체의 관계를 설정하고, 유스케이스의 기본 플로우만 분석함으로써 시퀀스 다이어그램을 정제한다.
④ 시퀀스 다이어그램을 완성한다.
⑤ 각 객체에 대해 추가되는 정보를 상세하게 문서화한다.
⑥ 새로운 유스케이스에 대해 다시 시작한다.



[그림 1] 객체 모델로의 유스케이스 매핑

- 시퀀스 다이어그램 작성 시의 문제점
  - 이벤트를 어느 레벨까지 낮추어야 할지 결정하기가 어렵다.
  - event decomposition 에 대한 것이 없다
  - 리턴 값이 있을 때, 문맥이 어렵고 어떻게 표현해야 할지 혼동스럽다.

(5) 객체 추출

시퀀스 다이어그램은 특정 유스케이스에 대한 이벤트 중심의 플로우로 객체간의 상호작용을 그래픽 형태로 표현해주기 때문에 클래스를 쉽게 식별할 수 있도록 해주며, 시스템의 프레임워크와 패턴을 발견하기 위한 수단으로도 사용된다.

정제된 시퀀스 다이어그램의 결과를 토대로 [표 3]과 같이 객체를 타입 별로 엔티티(Entity) 객체, 인터페이스(Interface) 객체, 컨트롤(Control) 객체로 구분하여 추출한다.

객체 타입	정 의
엔티티 객체	도메인 또는 비즈니스 객체를 의미하며, 시스템이 책임을 갖고 지속적으로 보관할 필요가 있는 객체
인터페이스 객체	시스템과 상호작용 하는 외부시스템에 대한 화면 인터페이스 객체
컨트롤 객체	비즈니스 규칙 또는 비즈니스 프로세스를 정의한 객체로, 다른 객체를 제어하거나, 엔티티도 인터페이스도 아닌 객체

[표 3] 객체 타입 별 정의

(6) 클래스 정의 및 클래스 다이어그램 작성

추출된 객체를 분석하여 시퀀스 다이어그램에서 정의한 객체의 메소드에 속성을 추가하여 클래스로 정의한다. 그리고 클래스간의 관계를 추출하여 전체 클래스 다이어그램을 완성한다.

- 클래스 다이어그램 작성 시의 문제점
  - 관계설정이 어플리케이션 도메인에 따라 다르게 표현될 수 있다
  - 구체화 시키면 클래스가 많아지는데, 클래스가 많아지면 관계 표현이 복잡해져 보기 힘들다

5. UML 의 문제점 조사

UML(Unified Modeling Language)은 서로 다른 언어들의 특징을 통합하여 산업표준으로 사용하기 위한 목적으로 Booch, Rumbaugh, Jacobson 에 의해 만들어진 모델링 언어로서, 분석 설계를 수행하는 방법론과 무관하게 UML 로 결과를 나타낼 수 있다. 따라서 UML 을 적용한 다양한 방법론들이 개발되어 왔다[6].

그러나 UML 은 객체지향 프로그래밍 언어로부터 하향식으로 진화된 것이어서 시스템의 이론적인 기초가 부족하다. 또한 지나치게 많은 다이어그램 타입과 프로그램 언어들의 영향을 받은 심볼들로 인해 모델이 복잡하며, 행위 모델링이 혼동스럽다는 문제점을 갖는다. 이러한 UML 을 복잡한 소프트웨어 시스템의 진화에 맞추어 시스템 아키텍트(Architect)의 도구로 선택되기 위해서는 보다 단순하고 사용자 친화적인 단일 모델로 구조와 행위를 통합해야 된다. 그러나 이해당사자(Stakeholder)들의 저항이 있기 때문에 대변혁의 개정은 어려운 실정이다[7].

분산시스템에서의 유스케이스 모델과 시스템 분해와 관련된 UML 의 문제점을 다음과 같이 분류된다[8].

- 비 본질적인(Accidental) 결점
  - 시스템과 액터 사이의 상호작용을 초기화해야 하는 시스템에서 유스케이스 모델은 상호작용 요구사항을 기술하기 어렵다.
  - UML 은 다양한 시스템 컨텍스트(context)를 모델링 할 수 없다.

→ 스테레오타입과 액터 사이의 연관성(association)을 허용하는 등의 간단한 수정으로 해결 가능

- 본질적인(Essential) 결점
  - 유스케이스 사이의 구조를 쉽고 직관적인 계층적 구조로 표현하기 어렵다.
  - 유스케이스의 상호작용을 처리하기 위한 적당한 수단을 제공하지 않는다.
  - 상태-의존적인 시스템의 행위 같은 상위 레벨 시스템의 행위를 모델링 할 수 없다.

→ 언어의 주요 개념에 영향을 미치는 수정이 필요

- 근본적인(Fundamental) 결점
  - 서브시스템들로 구성된 시스템에서의 모델링

정보에 대한 플로우를 다루기 힘들다.
- 분산시스템의 분해를 서브시스템으로도 UML 으로도 모델링 할 수 없다.
- 단일 관점으로 통합된 서브시스템과 같이 복합 구성된 엔티티의 모든 관점을 모델링 할 수 없다.

→ 언어의 기본적인 개념을 수정하더라도 제거되지 않음

이처럼 UML 의 문제점을 분석한 다양한 연구들이 있었으며, UML 1.x 의 공통적인 문제점을 살펴보면 거대한 사이즈, 불필요한 복잡성, 부정확한 의미, 비표준적인 구현, 제한된 커스터마이징, 컴포넌트 기반의 개발을 위한 지원이 부적절, 모델 다이어그램들을 교환할 수 없는 등의 문제점들이 지적되어 왔으며, UML 의 사이즈와 복잡성을 줄이기 위한 방안으로 개정이 진행되었다[9]. 그러나 새로이 개정된 UML2.0 역시 다음과 같은 기술적이고 정치적인 문제점이 있음이 지적되었다[10].

- 유스케이스는 사용자의 요구사항을 정의하기 위해 공통적으로 사용되는 것으로, 1.x 버전과 마찬가지로 다른 언어와 잘 통합되지 않는다.
- 내부구조를 갖는 컴포넌트와 클래스들 사이의 중요한 형식과 의미의 중복이 존재한다.
- UML2.0 에서의 많은 구조들은 하위레벨의 구조로서 통합되지도 증명되지도 않았다.
- UML 의 하부구조(infrastructure)는 불필요하게 복잡하고 관리하기 어렵다.
- 클래스의 상속구조는 정교하면서 거칠기(rough) 때문에 이해하고 관리하기 어렵다.

위에서 살펴본 바와 같이 UML 은 모델링 언어의 표준으로 자리잡고 있으면서도 많은 문제점을 가지고 있으며, 이러한 문제점들을 해결하기 위한 다양한 방안들이 제시되며 진화하고 있다. 그러나 UML 의 구조적인 문제점으로 인하여 UML 은 여전히 많은 문제점을 내포하고 있다 할 수 있다. 따라서 UML 의 문제점을 잘 파악하여 비즈니스 시스템 분석 시 고려해야 하며, 아울러 문제점을 보완할 수 있는 방법에 대한 연구도 필요하다.

6. 결론

본 논문에서는 사용자의 요구사항 분석을 위한 UML 기반의 유스케이스 모델링의 절차와 문제점에 대하여 살펴보았다. 또한 UML 의 전반적인 문제점들에 대하여 조사하였다.

현재 가장 많이 사용되고 있는 객체지향 개발 방법론과 CBD 방법론의 프로세스는 설계의 기본 모델링 언어인 UML 을 기반으로 하고 있으나, UML 의 복잡성과 분석 모델링에서의 구체적인 지침이 없기 때문에 경험자에 의존하여 분석하고 있는 실정이다. 따라서 프로젝트 분석을 위한 모델링 적용 시 고려해야 될 사항들을 미리 점검해 봄으로써 분석 시 발생할 수 있는 시행착오를 어느 정도 줄일 수 있도록 지침이 될 수 있을 것으로 기대된다.

참고문헌

[1] Jonathan, “Agile Methodologies and the Post Software Engineering Era”, http://www.featuredrivendevelopment.com/node/639, January 2004

[2] Eduardo Santana de Almeida, Alexandre Alvaro, Daniel Lcredio, Antonio Francisco do Prado, Luis Carlos Trevelin, “Distributed Component-Based Software Development: An Incremental Approach”, IEEE, 28th Annual International Computer Software and Applications Conference(COMPSAC’04), 2003

[3] John Torgersson, Alec Dorling, “Assessing CBD — What's the Difference?”, 28 th Euromicro Conference (EUROMICRO'02), September 2002

[4] Xia Cai, M.R. Lyu, Kam-Fai Wong, Roy Ko, “Component-based software engineering: technologies, development frameworks, and quality assurance schemes”, Seventh Asia-Pacific Software Engineering Conference (APSEC'00), December 2000, pp. 372

[5] Frank Armour, Granville Miller, “Advanced Use Case Modeling Software Systems”, Addison Wesley, 2000.

[6] José M. Fuentes, Víctor Quintana, Juan Llorens, Gonzalo Génova, Rubén Prieto-Díaz, “Errors in the UML Metamodel?”, ACM SIGSOFT Software Engineering Notes , Volume 28 , Issue 6, November 2003.

[7] Dov Dori, “Why Significant UML Change is Unlikely”, Communications of the ACM, November 2002/Vol. 45, No.11

[8] Martin Glinz, “Problems and Deficiencies of UML as a Requirements Specification Language”, IEEE IWSSD’00, 2000

[9] Cris Kobryn, “Will UML 2.0 Be Agile or Awkward?”, Communication of the ACM, Vol. 45,No.1, January 2002.

[10] Cris Kobryn, Expert’s voice UML 3.0 and the future of modeling, Published online, February 2004