

RUP 기반의 컴포넌트 시스템 설계

최창민*, 정정수*, 김천식*
*동명정보대학교 컴퓨터공학과
e-mail : yatamodu@tit.ac.kr

A Design On RUP based Component System

Chang-Min Choi*, Chung-Soo Chung*, Cheon-Sik Kim*
*Dept. of Computer Science, Tong-Myung University of Information Technology

요 약

오늘날의 컴퓨팅 시스템은 비즈니스 거래와 분산 업무 처리로 확대되어가고 있으며 정보 기술은 점차적으로 재사용성과 독립성 그리고 이식성을 가진 컴포넌트를 기반으로 한 응용 개발이 확산되고 있다. 컴포넌트 개발 형태는 코드의 재사용이나 클래스 라이브러리보다 좀 더 발전된 형태의 부품개발 형태로서, CBD(Component Based Development)를 기초로 한다. 그러나 CBD 를 이용하여 새로운 컴포넌트를 구축하는 비용의 증가와 함께 비즈니스 요구사항에 맞는 컴포넌트 개발을 위한 노력이 필요하며, 빠르고 정확한 컴포넌트 정보를 지원할 수 있도록 시스템 측면에서 정규화 형태의 컴포넌트 모델이 요구되고 있다. 본 논문에서는 사용자 요구사항에 접근하고, 재사용성과 독립성, 그리고 이식성을 가진 컴포넌트를 추출하고, 컴포넌트를 대상으로 인터페이스 나타내고자 한다. 따라서 본 논문에서는 대학 종합정보시스템 구축의 일부분인 구매관리 시스템을 분석 및 설계하여, 컴포넌트를 추출하고 컴포넌트 설계 모델을 제시한다.

1. 서론

환경이 복잡하고 빠르게 변화함에 따라서, 구축될 시스템 또한 제한된 기간에 저렴한 비용으로 원하는 시스템을 구축 해야 한다. 따라서 컴포넌트를 기반으로 한 시스템 구축은 필수적인 것이다. 이러한 컴포넌트 중심의 시스템 구축은 기능 중심의 독립적인 부품 단위의 컴포넌트를 조립하여 새로운 소프트웨어를 구축한다.

컴포넌트 기반의 소프트웨어의 개발은 품질향상, 빠른 개발과 유지보수의 효율성과 같은 소프트웨어 재사용에 의해서 이익을 얻을 수 있다. 따라서 이러한 마이너리 차원의 소프트웨어 컴포넌트 기반의 시스템 개발 중 가장 중요한 작업은 재사용성을 높일 수 있는 고 품질 컴포넌트의 효율적인 구축이다. 따라서 컴포넌트는 변화에 따라 쉽게 진화하거나 대체할 수 있도록 설계 되어야 한다.

본 논문에서는 객체 지향의 특성인 데이터와 기능의 통합, 캡슐화, 식별성을 기반으로 대체 가능성이 쉽도록 컴포넌트를 식별하고 명세화하기 위해 RUP(Rational Unified Process) 방법론을 이용하여 정의한다. 또한 이를 바탕으로 종합정보시스템의 한 부분

인 구매관리시스템을 적용하여 모델링 하고자 한다. 그리고 2 장에서는 CBD 개발방법론과 유즈케이스 중심의 분석에 대하여 알아보고, 3 장에서는 RUP 기반의 개발 모델 및 산출물에 관하여 알아보고, 4 장에서는 사례연구로 설계 모델링하며 5 장에서는 결론을 맺는다.

2. 관련연구

2.1 CBD 개발 방법론

소프트웨어 공학에서 중요한 기술적 변화로는 작업 라인 기반의 소프트웨어 개발에서 조립 가능하고, 구성가능하며, 그리고 독립적으로 추출된 컴포넌트들의 조합에 의해 만들어진 시스템을 제공하는 컴포넌트 기반의 소프트웨어 개발로의 전환이라 할 수 있다. 이렇게 개발된 시스템들은 그 시스템들의 변화로 인해 나머지 다른 시스템의 부품들에서 발생할 수 있는 문제를 최소화하였다.

컴포넌트란 잘 정의된 문제 도메인의 개념상에서 명확하게 제시된 기능들로 채워진 시스템의 독립적이고 대체 가능한 부품이라 할 수 있다. 소프트웨어 공학자들은 간단하게 컴포넌트들을 선택한 시스템 내부

에서 이들을 합성한다. 컴포넌트는 소프트웨어의 교체, 재사용이 용이하며 서비스는 인터페이스를 통해서만 이루어진다. 컴포넌트 개발의 각 단계는 개념적, 논리적, 물리적으로 각각 다른 형태를 취하고 있어 명확하며 이의 개발 대상은 비즈니스에서부터 어플리케이션, 통합소프트웨어에 이르기까지 다양하다. 또한 모두 연관성을 지니고 있기에 독립적인 패키지 형태로 작성이 되어도, 혹은 각각 다른 서비스를 제공하여도 이들이 모여 프레임워크를 구축하고 어플리케이션으로 발전한다.[1]

또한 컴포넌트를 조립해 새로운 어플리케이션을 만들 수가 있어 개발기간을 단축할 수 있으며, 기존의 컴포넌트를 재사용할 수 있다면 생산성과 경제성을 높을 수 있다. 사용자는 품질 좋은 컴포넌트를 선택하여 사용할 수 있으며, 컴포넌트의 사용은 인터페이스를 통해 이루어지고 실제 구현 과정은 사용자가 필요가 없다. 이러한 컴포넌트의 장점을 이용하고 체계적으로 개발하기 위해 CBD(Component Based Development) 방법론을 적용하여 컴포넌트를 만들고자 한다. CBD 에 있어서도 많은 방법론들이 있는데 대부분 UML 표기법을 따르고 있으며, 다양한 방법론 중 대중적인 방법론을 정리하면 <표 1>과 같다.[2]

<표 1> CBD 방법론

방법론	특징
Catalysis	<ul style="list-style-type: none"> • 흥미롭고 이론적인 접근법 • 너무 학문적이어서 사용하기가 어려움 • 실제화를 위한 프로세스 정리가 필요
CBD96	<ul style="list-style-type: none"> • Catalysis 를 근간으로 보다 명확하고 간단한 방법론을 제시 • UML 을 근간으로 하여 기존 객체지향 기법에 익숙한 소프트웨어 개발자는 쉽게 접근 가능 • 지원 도구들을 통하여 생산성과 품질을 최적화할 수 있는 장점 • 컴포넌트의 특징과 컴포넌트가 가져야 할 구성 요소들을 잘 정의, 새로운 방법론으로 확장이나 특정분야의 방법론으로 확장에 유용하게 사용
SELECT	<ul style="list-style-type: none"> • 이론적인 측면보다 실제적이고 개론적인 성향이 강해, 컴포넌트 기반 개발을 처음 접하는 개발자들의 컴포넌트 기반 개발 이해가 용이 • 정렬, 설계, 조립의 단계를 반복, 점진적으로 수행
RUP	<ul style="list-style-type: none"> • 객체지향 기법을 근간, 소프트웨어 개발에 필요로 하는 모든 관련 활동들을 총집합 • 대규모의 인원과 많은 시간을 필요로 하는 대형 프로젝트에 적합

2.2 RUP 방법론

RUP 는 요구사항 분석 단계의 산출물로 유스케이스 모델을 산출하고 분석 단계에서 각 유스케이스의 이벤트를 플로우를 보고 인터페이스 객체, 컨트롤 객체, 엔티티 객체를 도출해 나간다. 클래스 다이어그램을 도출할 경우에도 인터랙션 다이어그램을 통해서 효율적으로 도출한다. 또한 유사한 유스케이스를 그룹화

하여 시스템 컴포넌트인 패키지 다이어그램을 토출하고 패키지 다이어그램을 중심으로 클래스 다이어그램을 산출한다. 이는 시스템을 의미적으로 분할하여 클래스 다이어그램을 도출하므로 이해하기가 쉽고 분산 환경에서(Deploy)할 경우에 분할이 쉬우며 요구사항 변경시 요구사항 분석단계부터 설계단계까지 각 산출물을 추적할 수 있어 요구사항 변경을 쉽게 수용할 수 있다.

2.3 Catalysis

Desmond F.D' Souza 등에 의해서 제안된 Catalysis 는 1991 년에 시작된 OMT 의 정형화 작업에 그 뿌리를 두고 있다. Catalysis 는 비즈니스 모델로부터 소스 코드에까지 이르는 전 과정에서의 추적성을 보장하고, 불명확한 모델이나 문서를 최대한 명확하도록 만드는 정확성, 컴포넌트 기반의 개발, 코드뿐 아니라, 분석이나 디자인의 산출물들까지 확대된 재사용 개념, 확장성, 유연한 프로세스를 주요 특징으로 하고 있다.

Catalysis 는 각각 협력(Collaboration), 타입(Type), 정제(Refinement)의 3 가지의 모델링 개념에 기초하여 구성되어 있다. 협력은 연관된 객체들간의 상호연동으로 설명 할 수 있는데, UML 1.3 에서 Use Case 를 협력으로 실현된다. 협력은 특정 서비스를 제공해 주기 위하여 수행되는 작업들과 그 작업에 참여하는 객체들로서 이루어지는데, 협력은 큰 수준에서 점차 낮은 수준으로 세분화 될 수 있다. Catalysis 는 협력 모델을 통해서 대상 시스템 혹은 도메인의 기능들을 큰 수준에서 작은 수준으로 명세한다. Catalysis 방법론에서는 협력이 후보 컴포넌트의 기준이 되는 것이다.[3]

Catalysis 는 모델 요소와 프로세스 패턴 등의 다양한 좋은 기법들을 사용하고 있지만, 그 내용들의 많은 경험을 요구하고, 특히 컴포넌트 식별 작업의 경우, 그 절차가 개념적이고, 분석자의 직관에 많이 의존된다고 할 수 있다.

2.4 CBD96

CBD96 은 Computer Associates 사의 개발 도구인 COOL 을 이용한 컴포넌트 개발 아키텍처 모델이면서 개발 방법론이다. 컴포넌트의 식별은 시스템의 전체 도메인을 중심으로 타입 다이어그램과 비즈니스 타입 다이어그램을 추출하고 추출된 비즈니스 타입 다이어그램의 타입 중 핵심 타입(Core Type)을 중심으로 관련된 타입을 그룹화하여 비즈니스 컴포넌트를 추출한다. 이렇게 비즈니스 컴포넌트를 추출한 후 인터랙션 다이어그램(Interaction Diagram)을 통해 보다 독립적인 비즈니스 컴포넌트를 추출하도록 정제하는 과정을 거친 후에 비즈니스 컴포넌트를 조합하여 컴포넌트 아키텍처를 정의한다.

2.5 유스케이스 중심의 분석

유스케이스 중심의 분석상에서 기본적인 개념은 액터(Actor)와 유스케이스이다. 액터는 시스템의 사용자에 의해 수행되는 특정한 역할을 말하는 것으로 시스

템을 이용할 때 비슷한 행위를 하는 사용자들의 범주를 나타낸다. 유즈케이스는 시스템의 도메인에서 사용되어지는 언어들을 이용해서 자연어로 기술된다. 이러한 액터와 유즈케이스의 명세화를 통해서 유즈케이스 모델을 얻게 된다.

유즈케이스 중심의 분석은 요구사항 분석과정의 복잡성을 다루는 데에 도움을 준다. 액터의 사용적 측면에서 다른 유즈케이스들에 대해 독립적으로 발견하고 분석함으로써 하나의 유즈케이스에 초점을 둘 수 있기 때문이다. 유즈케이스 중심 분석의 개념은 간단하고 유즈케이스에 기술한 내용이 시스템의 문제 도메인에서 쉽게 발견되어 질 수 있기 때문에 고객과 시스템의 사용자가 의욕적으로 존재할 수 있는 사용자를 실제로 그들이 지닌 요구들과 행위들을 발견할 수 있다.

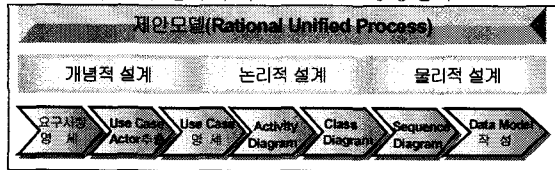
하지만 특정 유즈케이스는 모든 경우에 발생될 수 없다. 각 유즈케이스는 어느 특정 상황에서 시작되고 성공적으로 완수 되는 것을 기술하기 위한 것이다. 이러한 이유는 유즈케이스를 이용한 분석에서 다루어지지 않는다. 일반적으로 유즈케이스를 이용한 분석에서는, Jacobson의 A Use Case Driven Approach에서 정의된 바와 같이, 다음과 같은 이슈들에 대해서 충분히 다루어지지 않고 있다.[4]

- 유즈케이스는 독립적이지 않다. 그들 간에 겹쳐질 수 있고 동시에 발생될 수도 있다. 또는 서로간에 영향을 미칠 수도 있다.
- 유즈케이스는 어떤 조건이 만족되었을 때에 발생된다. 상황을 발생시키고 종결시키는 상황을 가지고 있다.
- 유즈케이스의 추상화(Abstraction)의 레벨과 기술되는 문장의 길이는 임의로 정해지게 된다.
- 유즈케이스는 실제로 시스템을 사용하는 시나리오의 모든 가능한 부분들을 다룬다는 보장을 하지 못한다.

3. RUP 기반의 개발 프로세스 및 시스템 구조

3.1 RUP 기반의 개발 프로세스

컴포넌트를 효과적으로 개발하기 위해서는 (그림 1)과 같이 실제계에서 사용자가 필요로 하는 요구사항을 통해 각각의 분석단계에서 구체 한 후, 컴포넌트 설계에서 구현하여 컴포넌트를 생성한다.

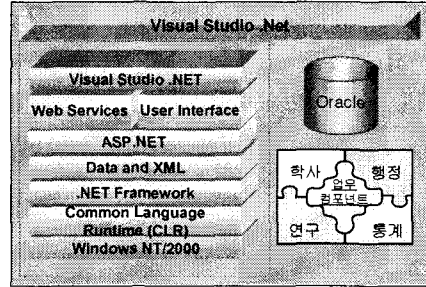


(그림 1) RUP 기반의 개발 프로세스

3.2 시스템 구조

종합정보시스템은 학사, 행정, 연구, 통계를 모든 사용자가 single-sign-on 을 통해서 전체의 시스템을 데이터 입력과 확인이 가능하며, 각각의 업무가 서로 연동되는 시스템을 구축한다. (그림 2)는 종합정보시

스템 구축의 시스템 구조이다.



(그림 2) 종합정보시스템의 시스템 구조

3.3 RUP 기반의 산출물

본 논문에서 요구사항 분석 모델링과 컴포넌트 설계 분석에서 필요한 산출물을 각 설계 단계에서 살펴볼 것이다. 분석단계에 따른 산출물은 <표 1>과 같다.

<표 1> 각 분석단계에 따른 산출물

구분	산출물
개념적설계	Use Case
	Use Case specification(명세서)
	Use Case Diagram Activity Diagram
논리적설계	Class Diagram Sequence Diagram
	물리적설계

3.3.1 개념적 설계 (Conceptual Design)

설계하고자 하는 업무에서 현재 사용자가 하는 일이 무엇이고, 비즈니스 요구사항이 무엇인지에 대해 이해하여 업무를 정의하는데 목적이 있다. 그러므로 실제 업무를 담당하는 담당자가 현 업무의 내용을 기술하는 것이 이상적이다, 그렇지 못할 경우 설계담당자가 요구자의 입장으로 설계를 진행한다 이 단계의 대표적인 산출물은 Use Case specification (명세서)이다.

Use Case specification 에는 개발하고자 하는 업무에 대한 모든 업무절차와 관련사항을 빠짐없이 기술하는 것이 핵심요소이다. 이 때, AS-IS 뿐만 아니라 TO-BE 의 업무 내역을 같이 기술할 수 있다. 기술 형식은 설계서를 나누던지, 아니면 한 설계서에서 변경되는 TO-BE 의 모습을 구분하여, 유연하게 정의하여 기술하도록 한다.

3.3.2 논리적 설계(Logical Design)

논리설계(Logical Design)는 개념설계(Conceptual Design) 단계에서 나열된 업무들을 대상으로 하여 누군가 수행해야 할 역할 및 그에 따른 서비스들을 정의하는 것이다. 이 설계단계는 시스템 설계자의 입장으로 이루어져야 한다. 개발할 시스템의 객체구조를 정의 하고, 정의된 객체간의 관계를 기술하고, 객체간의 커뮤니케이션 및 인터페이스를 정의하는 데 목적이 있다.

3.3.3 물리적 설계(Physical Design)

물리적 설계는 논리적설계에서 작성된 논리적

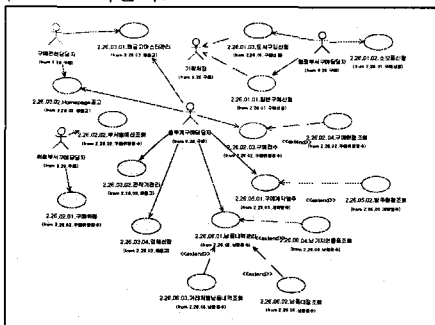
Analysis Model 을 대상으로 하여 실제 운영환경에서 나타날 수 있는 각종 문제점들을 사전에 찾아내고, 이에 대한 대책을 반영함으로써, 운영을 위한 최적의 물리적 Component Model 을 완성해 나가는 과정이라 할 수 있다. 이 단계는 설계자와 더불어 실 개발자가 같이 참여하여, Logical Design 내용을 이해하고, 개발자가 인지하고 있는 현 개발환경의 제약사항(기술구조, network, 시스템 Resource 등)하에서 발생할 수 있는 문제점에 대한 대책을 사전에 반영할 수 있도록 하는 것이 좋다.

4. 사례연구

본 논문에서 제시한 RUP 방법론을 적용하여 구매관리 시스템의 일부 컴포넌트를 사례연구로 설계 모델링 하였다.

4.1 구매관리시스템의 요구사항 분석

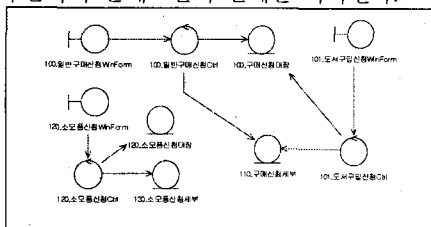
(그림 3) 은 구매관리시스템의 요구사항을 분석한 유즈케이스 다이어그램이다. 이 유즈케이스 다이어그램은 각 부서에서 하는 업무와 업무와 관련된 액터를 전체적으로 보여준다.



(그림 3) 구매관리시스템의 유즈케이스 다이어그램

4.2 구매관리 시스템의 비즈니스 모델링

(그림 4)는 구매관리 시스템의 구매신청 비즈니스 모델을 명시한 클래스 다이어그램이다. 이 클래스 다이어그램은 각각의 업무에 대하여 화면명세에 대칭되는 boundary 클래스와 데이터를 직접 호출하는 control 클래스, 그리고 데이터 테이블에 대칭되는 entity 클래스를 추출하여 클래스간의 관계를 나타낸다.

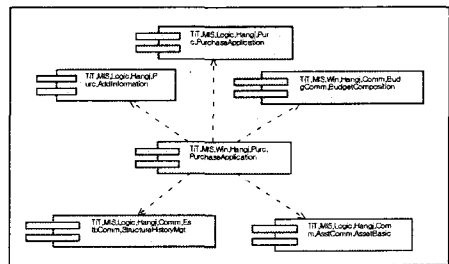
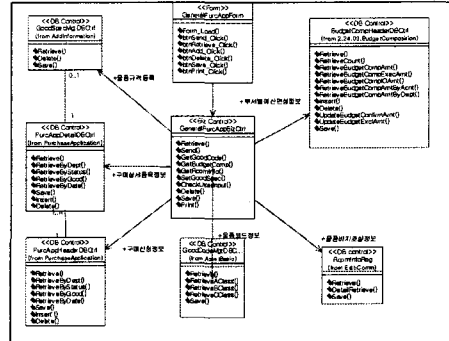


(그림 4) 구매신청 클래스 다이어그램

4.3 구매신청 시스템의 컴포넌트 인터페이스 식별

(그림 3)에서 “구매신청을 한다”의 유즈케이스 명세서의 기본 플로우를 바탕으로 시스템 계층의 컴포넌

트의 인터페이스를 식별하는 과정은 (그림 5)와 같다.



(그림 5) 컴포넌트 인터페이스 식별

위의 컴포넌트 다이어그램은 구매신청을 하기 위해서는 부가정보, 예산정보, 시설정보와 자산정보 컴포넌트를 참조하여야 함을 보여준다.

4.4 평가

종합정보시스템 구축의 일부분인 구매관리에서 구매신청에 대한 요구사항 분석 및 설계 후 컴포넌트를 도출 하였다. 이러한 과정을 통하여 전체 종합정보시스템의 컴포넌트를 식별하여 소프트웨어의 개발의 품질향상 및 빠른 개발과 유지보수의 효율성과 같은 소프트웨어 재사용에 의해서 이익을 얻을 수 있다.

5. 결론

RUP 기반의 개발 방법론을 유지하며, 개념적, 논리적, 물리적 설계를 진행하였다. CBD 기반으로 진행된 설계는 유즈케이스 크기에 종속적일 수 있다. 따라서 우선적으로 유즈케이스가 명확하게 도출 되어야 하며, 유즈케이스 간의 공통적으로 포함하는 기능을 컴포넌트로 식별하고 명세화 하여야 한다.

그리고 이후 다양한 영역에 적용할 수 있는 명확한 지침이나 보완에 대한 좀 더 상세한 연구가 필요하다.

참고문헌

[1] Bulter Group, What is a component, Interact, 1998.
 [2] 시사컴퓨터, “조립식 개발방법론 CBD”, 시사컴퓨터 기술보고서, 2001
 [3] Souza, Desmond, Objects, Component, and Frameworks with UML, Addison Wesley, 1999
 [4] Jacobson, I., et al, Object-Oriented Software Engineering: A Use Case Driven Approach, Addison-Wesley, 1992