제 19 회 한국정보처리학회 춘계학술발표대회 논문집 제 10 권 제 1 호 (2003. 5)

# CBD 기반 컴포넌트 리파지토리 시스템 설계 및 구현

박창섭, 연승호, 이해규, 박현규, 이상수
KT 서비스개발연구소 플랫폼연구팀
e-mail : {cspark0, shyeon, hkrhee, autosys, ssllee}@kt.co.kr

# Design and Implementation of a Component Repository System based on a CBD Methodology

Chang-Sup Park, Seung-Ho Yeon, Hae-Kyu Rhee, Hyun-Gyu Park, Sang-Soo Lee
Platform Research Team, Service Development Laboratory, KT Corporation

요 약

본 논문에서는 KT 서비스개발연구소에서 개발된 컴포넌트 리파지토리 시스템의 설계 및 구현 방법을 소개한다. 본 시스템은 CBD 방법론 및 EJB 컴포넌트 모델을 적용하여 유연하고 확장성 높은 컴포넌트 기반 시스템으로 개발되었다. 본 시스템은 컴포넌트를 기술하고 검색하기 위한 컴포넌트 명세 방법과 컴포넌트들을 체계적으로 분류, 관리할 수 있는 계층적 분류 체계를 정의한다. 또 컴포넌트의 재활용을 위해 효과적인 검색 및 탐색 방법을 제공하며, 사용자 관리 및 통계 기능 등을 포함한다. 본 논문에서는 요구사항 분석, 설계, 구현 단계에서 CBD 방법론의 적용 방안을 기술하고, 특히 컴포넌트 식별 및 컴포넌트 구조 설계 방법에 대해 상세히 기술한다.

## 1. 서론

최근 금융, 통신, 공공 등 여러 산업분야에서 소프트웨어 컴포넌트를 활용하여 새로운 응용 시스템을 개발하는 컴포넌트 기반 개발(Component-Based Development: CBD) 방법이 점차 확산되고 있다. 이 방법은 기 개발된 소프트웨어 자산을 재활용할 뿐만 아니라, 시스템을 컴포넌트 단위로 설계, 개발함으로써 생산성을 높이고 소프트웨어의 품질과 신뢰성을 향상시킬 수 있는 장점이 있다. 또 시스템의 수정 및 확장을 유연하게 하여 유지보수 비용을 줄일 수 있고 표준화된 구현 모델을 통해 호환성을 높일 수 있다[1].

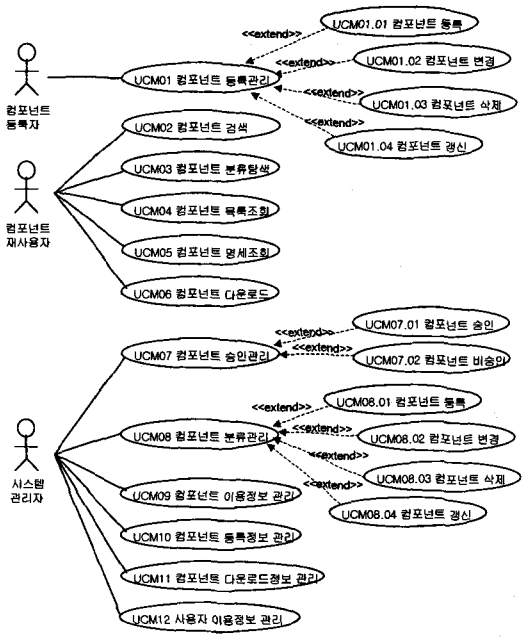
컴포넌트는 독립적으로 배치 가능하고 실행 가능한 소프트웨어 단위로서, 인터페이스(interface) 및 고객화 가능한 속성(property)들을 통해 외부의 컴포넌트나 타 시스템과 연동한다. 컴포넌트는 인터페이스와 구현을 분리함으로써 그 내부 구조 및 구현 방식을 고려하지 않고 쉽게 이용하거나 교체할 수 있다.

컴포넌트 기반 개발을 효과적으로 수행하기 위해서는 방법론뿐만 아니라 컴포넌트의 개발, 조립 및 배치를 위한 표준 개발 운용 환경과 기 개발된 컴포넌트들을 체계적으로 저장, 관리하고 검색하기 위한 리파지토리(repository) 시스템이 필요하다. 본 논문에서는 CBD 방법에 의한 컴포넌트 리파지토리 시스템의 설계 및 구현 방법에 대해 기술한다. 본 시스템은 요구사항 분석, 설계, 구현, 테스트 등 개발 전 단계에 KT 서비스개발연구소의 UML 기반 CBD 방법론인 KT-CBD 를 적용하여 개발되었다[2]. 본 시스템은 사용자 인터페이스, 비즈니스 로직, 데이터 관리 등으로 구분된 3-계층 구조를 갖으며, J2EE 의 엔터프라이즈 자바빈즈(Enterprise JavaBeans: EJB) 컴포넌트 모델을 이용하여 구현되었다. 본 시스템은 컴포넌트에 대한 명세 및 분류 방법, 효과적인 검색 및 탐색 방법, 사용자 관리 및 통계 기능 등을 제공한다.

본 논문은 다음과 같이 구성된다. 2 장에서는 컴포넌트 관리 및 리파지토리 시스템에 대한 기존 연구들을 살펴본다. 3 장, 4 장, 5 장에서는 본 시스템의 개발 프로세스 중 CBD 방법론에 따른 요구사항 분석, 컴포넌트 및 시스템 설계, 그리고 시스템 구현 방법에 대해 기술한다. 마지막으로 6 장에서 결론을 맺는다.

## 2. 관련 연구

최근 학계 및 산업계에서 CBD 를 지원하기 위한 컴포넌트 리파지토리 시스템에 대한 연구 및 프로토타입(prototype) 시스템 구축이 이루어 지고 있다[3].




[그림 1] 유스케이스 도

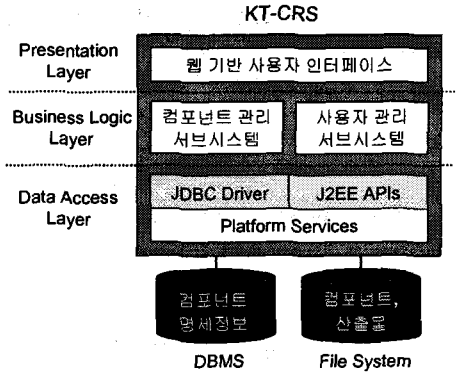
R. Meling 등은 소프트웨어 컴포넌트의 검색 방법을 제안하고 컴포넌트의 효과적인 관리를 위해 도메인 정보와 의미 정보를 통합하여 컴포넌트를 기술할 수 있는 컴포넌트 명세 및 분류 방법을 정의하였다[4]. J. Cha 등은 소프트웨어의 계층적 컴포넌트 구조 및 컴포넌트 명세 방법을 정의하고 컴포넌트 메타 데이타를 분류하였으며, 컴포넌트 리파지토리 프로토타입 시스템을 구축하였다[5]. [6]에서는 컴포넌트 저장소의 요구사항을 분석하고 J2EE 응용 모델을 바탕으로 웹 기반의 컴포넌트 저장소를 구현하였다.

한편, Adaptive 사와 Flashline 사 등에서는 상용 컴포넌트 관리 솔루션을 제공하고 있다. Adaptive *Component Manager* 는 프로그램, 텍스트 및 XML 문서, 이미지 파일, 설계 문서 등을 포함한 컴포넌트 관리를 위한 웹 기반 솔루션을 제공한다[7]. 이 시스템은 컴포넌트의 저장, 분류, 검색, 재사용, 문서화, 프로젝트 협업 기능 등을 제공하며, *Unisys Universal Repository* 를 내부 저장시스템으로 사용한다. Flashline 사의 Component Manager Enterprise Edition 은 다양한 소프트웨어에 대한 체계적인 분류와 재사용을 지원하는 컴포넌트 관리 솔루션으로, J2EE, .NET, XML, 웹서비스 등의 유형을 지원하고, SOAP 기반 API 를 제공한다[8].

## 3. 요구사항 분석

### 3.1 기능

본 시스템의 사용자는 컴포넌트 등록자, 컴포넌트 재사용자, 그리고 시스템 관리자 등으로 구분된다. 컴포넌트 등록자는 개발된 컴포넌트를 저장, 변경, 갱신 및 삭제할 수 있다. 컴포넌트 재사용자는 새로운 소프트웨어의 개발을 위해 저장되어 있는 컴포넌트를 검색, 조회 및 다운로드할 수 있다. 시스템 관리자는 컴포넌트 분류 체계 관리, 사용자 관리, 이용 통계 생성 등을 포함한 시스템 관리를 수행한다.




[그림 2] 시스템 개념도

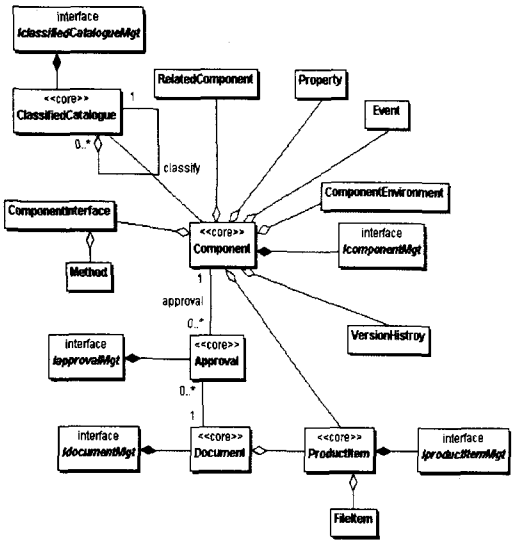
그림 1 은 시스템에 대한 행위자(actor)와 기능적 요구 사항들 사이의 관계를 나타낸 유스케이스(Use-Case)도의 일부이다. 각 유스케이스에 대해서 상세 유스케이스 시나리오가 작성되며, 이러한 유스케이스 모델을 바탕으로 시스템의 세부 기능이 정의된다[9].

### 3.2 컴포넌트 명세

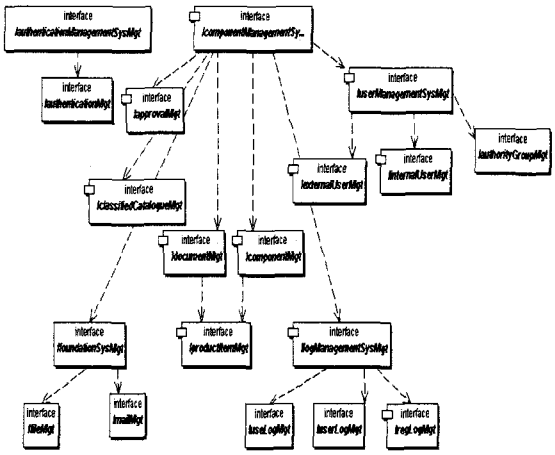
컴포넌트들의 효과적인 관리 및 재활용을 위해서는 컴포넌트의 사양, 기능, 특성 등을 효과적으로 기술하기 위한 명세 방법이 필요하다. 본 시스템에서는 기본 명세정보, 기능정보, 기술정보 등으로 구분되는 메타 정보들을 통해 컴포넌트를 정의한다. 기본명세정보는 컴포넌트의 사양을 기술하는 정보들로서, 이름, 버전, 개발자, 분류 정보, 제작일, 등록일, 관련 컴포넌트, 라이센스, 키워드, 설명 등을 포함한다. 기능정보는 컴포넌트의 기능을 기술하는 정보로서, 컴포넌트의 인터페이스, 오퍼레이션, 고객화 가능한 속성 및 이벤트 정보 등을 정의한다. 기술정보는 컴포넌트의 구현 및 사용과 관련된 기술적, 환경적 정보들로서, 개발언어, 개발도구, 실행가능한 HW, OS, 필요한 미들웨어, DBMS, 그리고 기타 물리적 요구사항 등으로 구성된다.

### 3.3 컴포넌트 분류 체계

본 시스템에서는 저장된 컴포넌트들을 효과적으로 관리하기 위해 트리 형태의 계층적 분류 체계를 제공한다. 분류 트리의 각 노드는 분류 기준 속성의 특정 범주를 나타내며, 자식 노드들은 부모 노드를 세분화한 범주들이다. 컴포넌트는 분류 트리 내의 임의의 노드에 속할 수 있다. 이러한 계층적 분류 체계는 컴포넌트들을 의미적(semantic)인 특성에 따라 체계적으로 분류하고 사용자가 등록된 컴포넌트를 탐색하고 검색하는데 이용된다. 본 시스템은 컴포넌트를 사용 영역, 유형, 구현 기술 등의 기준으로 분류할 수 있는 분류 트리 집합을 제공하고, 시스템 관리자가 이들을 생성, 변경 및 삭제할 수 있는 기능을 제공한다.




[그림 3] 비즈니스 인터페이스 책임도




[그림 4] 초기 인터페이스 의존성도

## 4. 시스템 설계

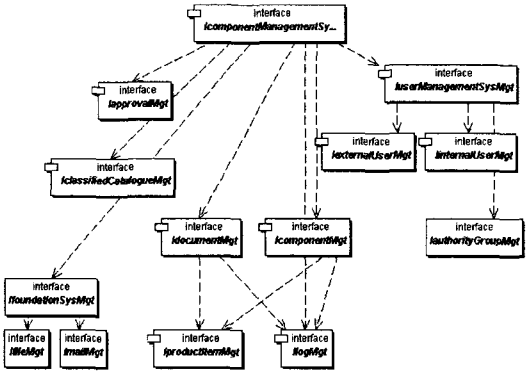
본 시스템은 유연성과 확장성을 위해 그림 2 와 같은 3-계층 구조를 갖는다. 시스템의 주요 기능들은 비즈니스 로직 계층에서 컴포넌트로 구현되며, 크게 컴포넌트 관리 서브시스템과 사용자 관리 서브시스템으로 구분된다. 본 장에서는 CBD 방법론을 적용한 시스템 설계 방법에 대해 기술한다. 특히 컴포넌트 관리 서브시스템을 중심으로 분석 단계에서 작성된 유스케이스 모델로부터 컴포넌트를 식별한 후 컴포넌트들 간의 상호작용 분석을 통해 컴포넌트 구조를 도출하는 과정에 대해 상세히 설명한다.

### 4.1 컴포넌트 식별

일반적으로 응용 시스템을 구성하는 컴포넌트는 외부에서 시스템의 서비스에 접근하기 위한 인터페이스를 제공하는 시스템 컴포넌트와 시스템의 기능, 즉 비즈니스 로직을 구현하는 비즈니스 컴포넌트로 구분된다.

시스템 컴포넌트를 정의하기 위해서는 먼저 요구사항 분석 결과인 유스케이스 모델을 기반으로 시스템 인터페이스를 식별한다. 시스템 인터페이스는 UI 계층과 비즈니스 로직 계층 사이를 연결하는 역할을 한다. 본 시스템에서는 각 서브시스템에 대해 하나의 시스템 인터페이스를 정의하였다. 시스템 오퍼레이션들은 유스케이스 시나리오에 대한 시스템 시퀀스도를 통해 식별한다. 각 시스템 인터페이스에 대해 일반적으로 하나의 시스템 컴포넌트를 생성한다.

비즈니스 컴포넌트를 식별하기 위해서는 먼저 시스템을 구성할 객체들을 추출하고 객체의 속성, 오퍼레이션, 객체들 사이의 연관 관계 및 다중성 등을 정의하는 객체 모델을 생성한다[9]. 객체 모델에서 핵심(core) 객체들을 도출하고 그것들로부터 비즈니스 인터페이스들을 식별한다. 각 핵심 객체 및 그것과



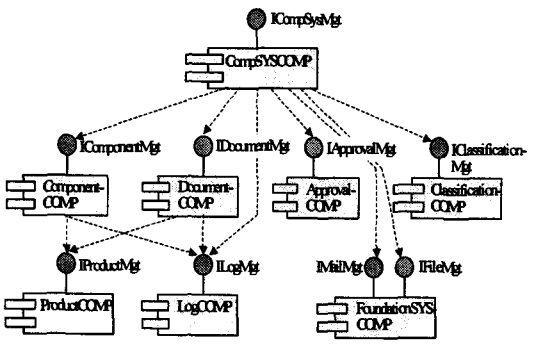

[그림 5] 정제된 인터페이스 의존성도

논리적 연관성이 있고 상호 참조가 많은 객체들은 하나의 비즈니스 인터페이스를 통해 접근되도록 설계된다. 그림 3 은 본 컴포넌트 관리 서브시스템의 객체들과 비즈니스 인터페이스들 사이의 연관 관계를 나타낸다. Component, Document, Product, Approval, ClassifiedCatalogue 등 5 개의 객체가 핵심 객체로 선정되었고 각각 하나씩의 비즈니스 인터페이스가 할당되었다. 이 결과를 바탕으로 초기 컴포넌트 구조가 설계된다[9].

### 4.2 컴포넌트 상호작용 분석

식별된 시스템 및 비즈니스 인터페이스들을 기반으로 각 유스케이스에서 인터페이스들 간의 상호작용을 분석하여 각 비즈니스 인터페이스가 담당할 비즈니스 오퍼레이션들을 구체적으로 정의한다. 그리고 인터페이스들 사이의 의존성을 분석하고 인터페이스들의 책임을 적절히 조정함으로써 최종 컴포넌트 구조를 결정한다.

그림 4 는 시스템의 초기 인터페이스 의존성도를 나타낸다. 이 모델에서 인증 관리 부분의 기능이 비교적 단순하므로 이를 사용자 관리 인터페이스에




[그림 6] 컴포넌트 구조도

포함시키고, 로그 관리 기능의 경우 컴포넌트 및 문서 인터페이스에 대한 의존성이 높으므로 하나의 인터페이스로 통합하여 이들 인터페이스와 직접적인 의존 관계를 갖도록 수정하였다. 그림 5 는 이러한 정제 결과를 나타낸다.

인터페이스 의존성 모델로부터 각 인터페이스에 대해 이를 구현할 컴포넌트를 정의한다. 일반적으로 1:1 관계를 갖으나, 의존성이 강한 인터페이스들은 하나의 컴포넌트에서 처리하는 것이 효과적일 수 있다. 본 시스템에서는 파일 처리와 메일 송신을 위한 두 인터페이스를 하나의 기초 컴포넌트로 구현하였다. 그림 6 은 본 컴포넌트 관리 서브시스템의 최종적인 계층형 컴포넌트 구조를 나타낸다.

### 4.3 컴포넌트 설계 및 시스템 구조 정의

확정된 컴포넌트들의 설계는 내부 클래스도와 내부 시퀀스도를 이용하여 컴포넌트의 속성 및 오퍼레이션을 세부적으로 정의한다. 시스템 구조는 컴포넌트 구조 외에 서버, DBMS, 하드웨어 등을 포함한 전체 시스템의 물리적 분산 구조 및 컴포넌트 배치(deployment) 구조를 포함하여 정의한다[2].

## 5. 시스템 구현

본 컴포넌트 리파지토리 시스템은 J2EE 플랫폼 상에서 EJB 컴포넌트 모델을 이용하여 구현되었다[10]. 비즈니스 로직 계층의 컴포넌트들은 EJB 객체들의 집합으로 구현되며, 사용자 인터페이스 계층은 자바서버 페이지(JSP) 기술을 이용한다. JSP 와 EJB 객체들 사이의 호출은 자바빈즈 객체를 매개로 하여 이루어 진다. 데이터 관리 계층에서 컴포넌트 명세정보는 JDBC 를 통해 데이터베이스 시스템에 저장 관리되며 컴포넌트 자체와 관련 산출물들은 자바 I/O API 를 이용하여 파일 시스템에 저장된다. 한편, 동시성 제어와 트랜잭션 처리, EJB 객체들의 생명 주기 및 시스템 자원의 관리는 EJB 서버에 의해 자동적으로 이루어 진다.

4 장에서 설계된 비즈니스 컴포넌트에 속한 객체들은 각각 하나의 EJB 객체로 구현된다. 즉, 하나의 비즈니스 컴포넌트는 여러 개의 EJB 객체들로 구성된 패키지로 구현된다. 각 EJB 객체는 EJB 스펙에 따라 세 개의 자바 객체, 즉 홈(home) 인터페이스, 리모트(remote) 인터페이스, 빈(bean) 객체로 정의되고 EJB 서버에 의해 구현된다. 이러한 EJB 컴포넌트들은 EJB 서버인 웹 응용 서버 상에 배치되어 실행된다.

EJB 객체들은 그 기능 및 성격에 따라 엔티티 빈과 세션 빈으로 구분된다. 비즈니스 로직이나 업무 흐름 처리, facade 인터페이스의 역할은 세션 빈으로 구현되고, 데이터베이스나 파일 시스템을 통해 지속성을 보장해야 하는 객체들은 엔티티 빈으로 구현된다. 본 시스템에서 각 컴포넌트들은 대부분 하나의 세션 빈과 하나 이상의 엔티티 빈들로 이루어 진다.

## 6. 결론

본 논문에서는 KT 서비스개발연구소에서 개발된 컴포넌트 리파지토리 시스템의 설계 및 구현 방법을 제시하였다. 본 시스템의 특징은 자체적인 CBD 방법론 및 EJB 컴포넌트 모델을 적용하여 유연하고 확장성 높은 컴포넌트 기반 시스템으로 개발되었다는 점이다. 본 시스템은 컴포넌트를 효과적으로 기술하고 검색하기 위한 명세 방법과 다양한 기준에 의해 체계적으로 컴포넌트들을 분류, 관리할 수 있는 계층적 분류 체계를 제공한다. 또 직관적이고 효과적인 검색 및 탐색 방법을 제공하며, 사용자 관리 및 통계 기능 등을 포함하고 있다. 본 시스템은 사내 소프트웨어 컴포넌트 자산의 공유 및 재활용을 통해 소프트웨어와 서비스 개발의 생산성 및 품질 향상에 기여하고 컴포넌트 기반 개발 방법을 지원하는데 이용될 수 있다.

## 참고문헌

[1] David Chappell, The Next Wave: Component Software Enters The Mainstream, Whitepaper, Rational Software Corp., http://www.rational.com/products/whitepapers/354.jsp, 1997.

[2] KT-CBD 방법론, KT 서비스개발연구소, 2002.

[3] J. Guo and Luqi, A survey of software reuse repositories, In Proc. of the 7th IEEE Int'l Conference and Workshop on Engineering of Computer Based Systems, pp.92–100, 2000.

[4] R. Meling, E.J. Montgomery, P. Sudha Ponnusamy, E.B. Wong, and D. Mehandjiska, Storing and retrieving software components: a component description manager, In Proc. of the 2000 Australian Software Engineering Conference, pp.107–117, 2000.

[5] J. Cha, Y. Yang, M. Song, and H. Kim, Design and implementation of component repository for supporting the component based development process, In Proc. of the 2001 IEEE Int'l Conference on Systems, Man, and Cybernetics, pp.735–740, 2001.

[6] 박윤필, 염근혁, J2EE 어플리케이션 모델을 따른 웹 기반의 컴포넌트 저장소 구현, 소프트웨어공학회지, 제 13 권, 제 4 호, pp.7–14, 2000.

[7] Adaptive, Ltd., http://www.adaptive.com.

[8] Flashline, Inc., http://www.flashline.com.

[9] 컴포넌트 리파지토리 시스템(KT-CRS) 개발보고서, KT 서비스개발연구소, 2002.

[10] R. Monson-Haefel, *Enterprise JavaBeans*, 3rd edition, O'Reilly & Associates, Inc., 2001.