사례 발표

Enterprise 포탈 구축 시 CBD 활용 사례

이 경 배[1] 박 준 성[2] 이 재 익[3] 강 금 석[4]

목 차

1. 배 경

1.1 프로젝트의 기술/업무적 특징

새롭게 구축된 기업용 포탈 시스템에 요구되었던, 기술/업무적 특징은 다음과 같다.

1.1.1 기술적 특징

- 확장성 : 모바일등 멀티 채널지원, 웹서비스 등으로 확장이 가능해야 한다.
- 유지보수성 : 기능 및 기술 구조 변경에 대해 민첩하게 대응할 수 있는 아키텍쳐야 한다.
- 안정성 : 14만 유저 이상의 피크타임 부하를 견딜 수 있어야 한다.

1.1.2 업무적 특징

- 접속성 : 사내 망 또는 인터넷을 통한 사외 망에서 접속 가능해야 한다.
- 보안성 : 사내 중요한 정보를 다룰 수 있으므로, 보안성이 높아야 한다.
- 글로벌 버전 :글로벌 오퍼레이션을 고려한 시스템 및 어플리케이션 구조여야 한다.

1.2 CBD 적용 목적 및 수준

본 프로젝트는 기존에 개발된 MySingle 2.0을 컴포넌트 중심의 시스템으로 변경하는 프로젝트였다. 기존 시스템을 컴포넌트화한 이유는 다음과 같다.

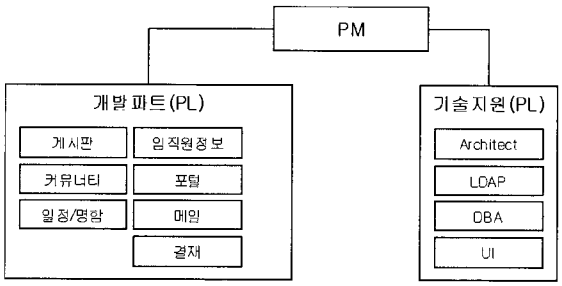
- 시스템 모듈별 또는 Layer 별 분산 배치를 용이하게 한다.
- 시스템 기능 및 기술 구조 변경에 민첩하게 대응하게 한다.
- 멀티 채널 또는 웹 서비스 등의 신규 기술 구조로의 시스템 확장이 용이하게 한다.
- 프로젝트 내 공통 부분에 대한 코드 재사용을 통해 품질 및 생산성을 향상시킨다.

본 프로젝트에서는 위와 같은 이유로 컴포넌트 중심의 시스템을 개발했기 때문에, 컴포넌트를 처음부터 재사용을 목적으로 설계/개발하는 Systematic Reuse 방식은 취하지 않았다. 따라서 개발된 컴포넌트들이 향후에 재사용되려면, 추

1) 삼성SDS 정보전략실 상무
2) 삼성SDS 정보전략실 상무, CTO 및 CKO (겸)첨단소프트웨어공학센터장
3) 삼성SDS 사내교수
4) 삼성SDS 정보기술연구소 책임연구원

가적인 공통성 및 가변성 분석을 통한 Refactoring이 필수 불가결하다. 단, 일부 기술 인프라스트럭쳐성 컴포넌트는 다른 프로젝트에서도 충분히 재사용될 수 있을 것으로 보인다.

1.3 추진 조직



(그림 1) 프로젝트 조직도

추진 조직은 크게 개발 파트와 기술지원 파트로 나뉘고, 개발 파트는 대표 모듈 중심으로 소그룹화하여 각각 모듈 리더(ML) 및 분석/설계자, 개발자를 배치했고, 기술 지원 파트는 아키텍트, 기술에 대한 전문가를 배치하였다. 개발파트는 업무의 효율화를 위해 모듈별 기존 시스템의 개발/운영 인력과 신규 인력 함께 배치하였다. 인원은 프로젝트 단계에 따라 변동이 있었으나, 개발 파트는 평균 40여명이었고, 기술 지원 파트는 10여명이었다.

2. 개발 환경

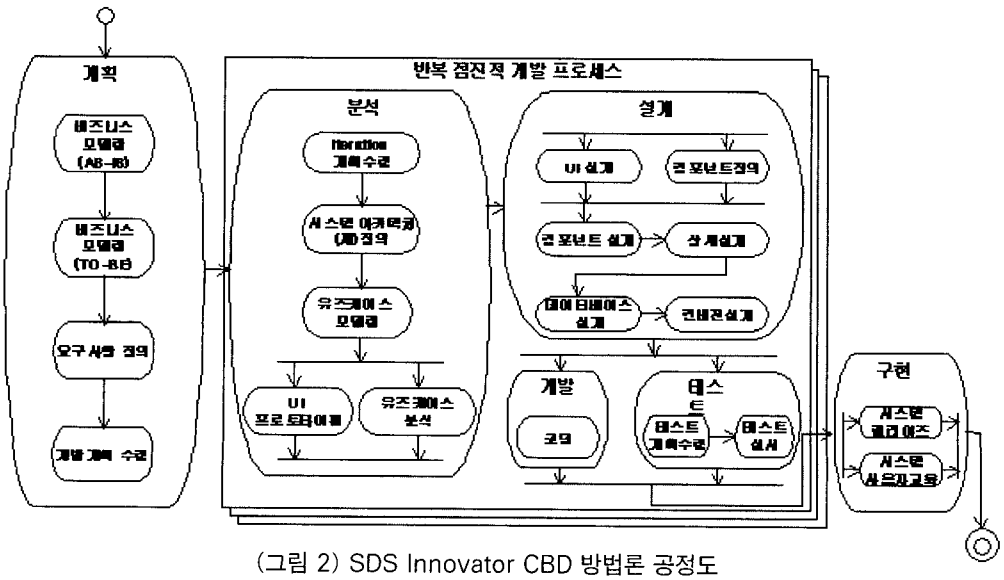
2.1 개발 방법론

2.1.1 SDS CBD 방법론

SDS Innovator CBD 방법론은 재사용이 가능한 컴포넌트의 개발 또는 상용 컴포넌트들을 조합하여 어플리케이션의 개발 생산성과 품질을 높이고 또한 시스템의 유지보수 비용을 최소화할 수 있는 개발방법 프로세스를 정의한다[1].
CBD방법론의 특징은 다음과 같다.

2.1.1.1 컴포넌트 기반 개발

독립적인 기능을 수행하는 컴포넌트 단위로 어플리케이션을 재사용이 가능한 단위로 분할하여 개발/조립하는 방식으로 시스템을 구축한다.



(그림 2) SDS Innovator CBD 방법론 공정도

2.1.1.2 표준화된 UML을 통한 모델링 및 산출물 작성

프로젝트의 전체 라이프사이클을 지원함으로써 일관성있는 모델링 및 산출물 작성이 용이하고, 국제 표준인 OMG의 UML 1.3 표준을 준수한다.

2.1.1.3 반복 점진적 개발 프로세스 제공

위험 요소를 초기에 파악하고 고객의 요구사항 변동에 유연하게 대응한다.

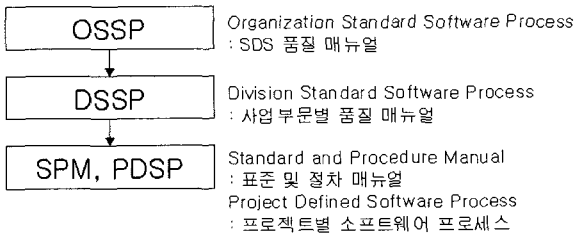
2.1.1.4 표준화를 통한 재사용성 향상

표준화된 산출물 작성 기법 및 표준화된 컴포넌트 제작 기법을 통해 재사용성을 향상한다.

2.1.1.5 아키텍처 중시

고객,개발자,의사 결정권자 등 모든 프로젝트 관련자에게 시스템 구조에 대한 일관된 다양한 View를 제공한다.

2.1.2. 프로젝트 정의 소프트웨어 프로세스

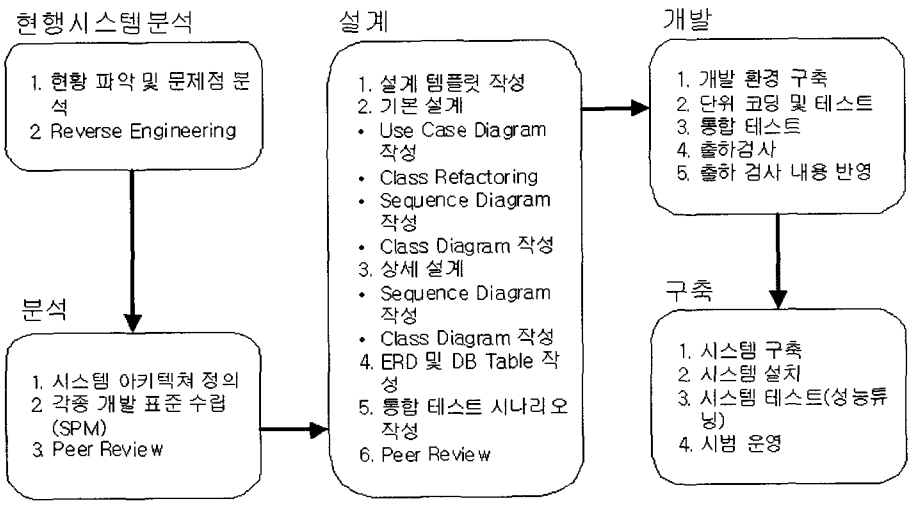


(그림 3) SDS 품질 매뉴얼 체계

SDS의 품질 매뉴얼 체계는 다음과 같이 이루어져 있다.

- SDS 품질매뉴얼 (OSSP) : 전사 공통의 표준 프로세스 정의
- 사업부문별 품질매뉴얼 (DSSP) : 사업부내 표준 프로세스 정의
- 표준 및 절차 매뉴얼(SPM) : 프로세스의 성격이 단위부서 및 프로젝트 내의 표준 프로세스 정의
- 프로젝트 정의 소프트웨어 프로세스(PDSP) : 프로젝트에 맞게 정의된 소프트웨어 개발 프로세스로, SPM 내에 포함됨

하위의 표준은 상위의 표준을 준수하도록 정의된



(그림 4) 프로젝트 정의 소프트웨어 프로세스

다. 본 프로젝트에서는 상위의 OSSP및 DSSP를 준수하고, SDS Innovator CBD 방법론을 커스토마이징해서, 아래와 같은 프로젝트 정의 소프트웨어 프로세스(PDSP)를 정의하여 사용하였다[2].

2.2. 개발자 역할 및 협업

본 프로젝트는 다음의 〈표 1〉에 정의한 역할을 가진 사람들로 구성되었다.

〈표 1〉 개발자 역할

역 할	역 할 설 명
프로젝트 관리자	프로젝트의 전반적인 진행에 대한 책임
QAO	시스템과 시스템 산출물에 대한 품질보증 활동 수행
분석/설계자	고객의 기능 및 비기능적인 요구사항을 분석 및 설계에 반영
아키텍트	소프트웨어 및 기술 아키텍쳐 정의 및 검증
UI 개발자	UI 레이어에 관련된 요소를 개발 및 단위 테스트
컴포넌트 개발자	컴포넌트 개발 및 단위테스트
어플리케이션 개발자	컴포넌트를 조립하여 서버 레이어 개발
테스터	개발된 시스템의 테스트 수행 및 결과 취합
시스템 관리자	개발 및 운영환경에 필요한 시스템을 운영 및 관리
도메인 전문가	해당 업무영역에 대한 전문지식을 가진 자
기술 전문가	개발 및 운영 환경에 사용되는 각종 상용 개발도구 및 기술에 대한 전문가

한 사람이 여러 역할을 수행하는 경우도 있었지만, 아키텍트 및 기술 전문가와 일반 개발자와의 분리, 개발자 내부도 컴포넌트 개발자, UI 개발자, 어프리케이션 개발자의 구분은 명확히 하였다. 다음은 대표 역할간의 협업에 대한 설명이다.

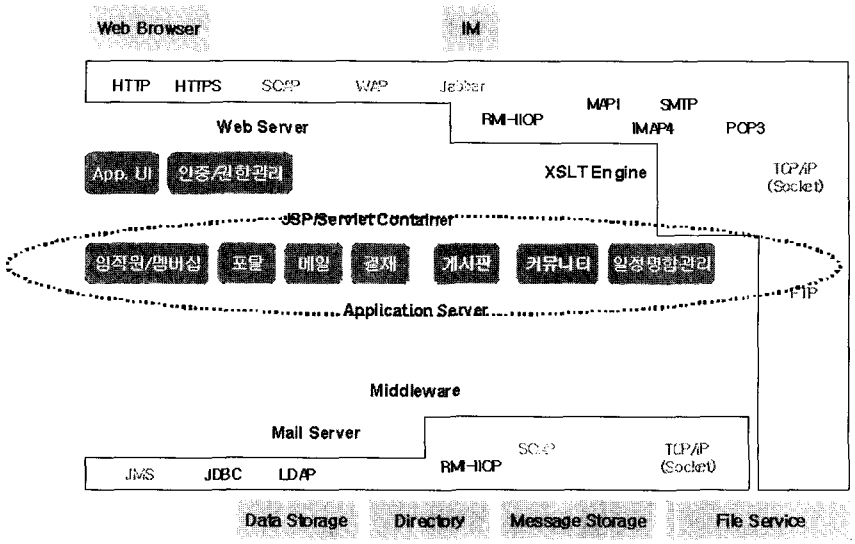
- 기존 개발인력은 기존 시스템의 Reverse Engineering을 통해 시스템 요구사항 및 시스템 내부 로직을 정의하는 분석/설계자 및 도메인 전문가 역할을 맡았다.
- 신규 인력은 UML 을 이용한 디자인 템플릿과 코드 템플릿을 제공하고 기존 인력과 같이 시스템 상세 설계 및 개발을 진행하는 기술 전문가 및 분석/설계자 역할을 맡았다.
- 개발자는 UI 개발자, 컴포넌트 개발자, 어플리케이션 개발자로 역할을 구분하여, 설계된 내용을 바탕으로 개발이 동시에 병력적으로 진행되도록 했다.
- 아키텍트는 시스템 또는 소프트웨어 분야의 플랫폼 및 각종 기술 표준을 수립하고, 전체 개발자들은 이 표준을 따르도록 함으로써, 기술에 관련된 의사소통이나 의사결정에 걸리는 시간을 최소화 시켰다.
- 기술 전문가는 사전에 사용될 기술 검토 및 템플릿을 제시하도록 함으로써, 설계된 내용을 제때 제대로 구현할 수 있도록 했다.
- 아키텍트와 기술 전문가 인력이 제품이 수행될 환경에 대한 성능 타당성 검토를 사전에 진행하여 위험 요소를 줄였다.

3. 시스템 아키텍쳐

3.1 EP 개발 Framework

본 프로젝트에서 적용한 EP 개발 Framework 은 다음과 같다.

향후, 어플리케이션 기능뿐만 아니라, 클라이언트 종류, 백엔드 시스템 등의 확장이 있을 때에도 사용 가능하고 어플리케이션 서비스는 이 Framework 상에서 자유로이 추가되는 형태로 설계되었다. 본 프로젝트에서는 엔터프라이즈 포탈을 위해 일차적으로 (그림 5)에서 보는 바와 같이 서비스를 제공하도록 개발되었다.

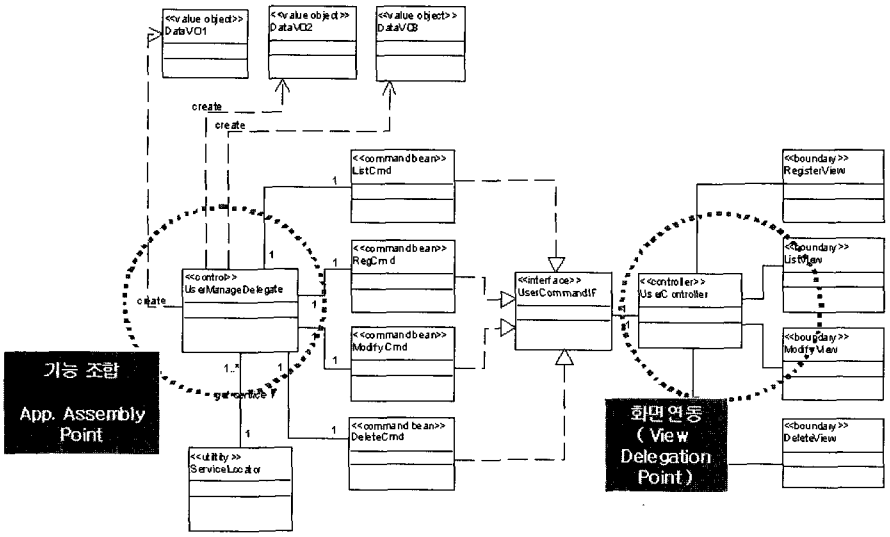


(그림 5) EP Framework

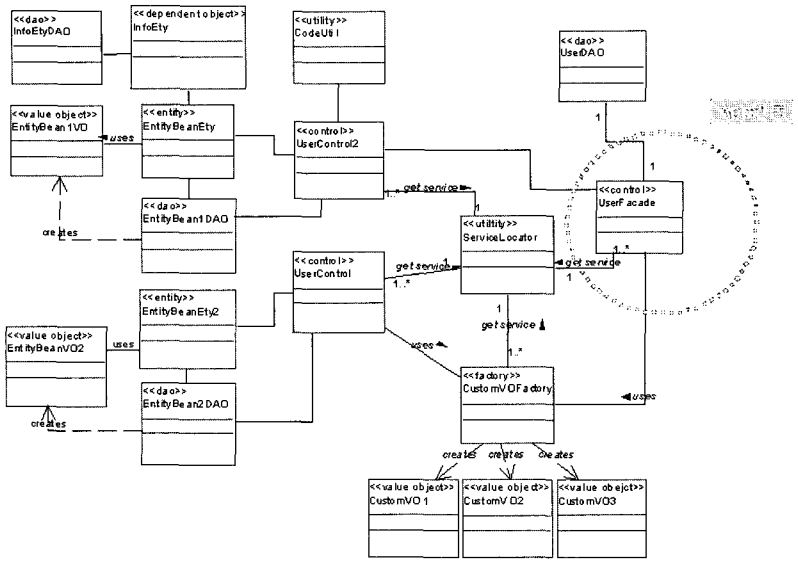
3.2 Web Tier

웹 계층은 주로 Presentation 계층을 구현하는 부분으로 사용된 디자인 Framework은 다음 (그림 6)과 같다.

웹 계층에서는 상호 연동할 수 있는 부분이 URL을 통한 화면 단위 연동 부분, 비즈니스 컴포넌트로부터 서비스를 조합할 수 있는 연동 부분을 가지고 있다. 이 두 부분을 통해 타 모듈의 서비스 이용 및 조합, 화면 서비스를 받을 수 있도록 설계되었다. 이 디자인 Framework적용은 각 모듈당 하나씩 적용하였다.



(그림 6) Design Framework for Web Tier



(그림 7) Design Framework for Application Tier 》

3.3. Application Tier

어플리케이션 계층을 위해 사용된 디자인 Framework은 다음과 같다.

모든 서비스 요청은 맨 앞 단의 Facade[3,4,5]에서 지정된 인터페이스를 통해 받도록 되어 있다. 이 디자인 Framework이 적용되는 단위는 Use Case Diagram에서 최하위 System Boundary 당 하나씩 적용하였다.

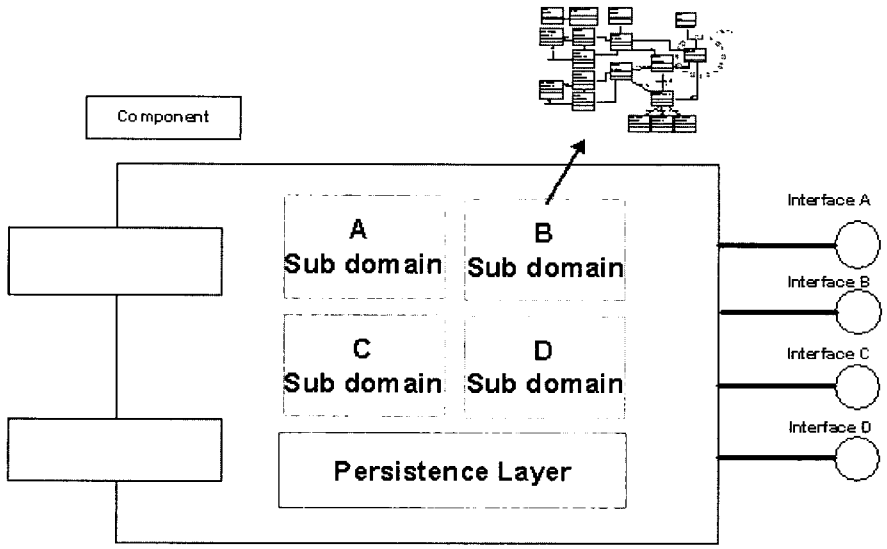
3.4 컴포넌트 Granularity

컴포넌트는 정의하기에 따라서 여러 가지 형태를 포함할 수 있으나, 실제 개발입장에서 의미가 있는 컴포넌트는 협의의 컴포넌트 의미를 가질 수 밖에 없다. 본 프로젝트에서는 컴포넌트를 재사용 가능하고 독립된 수행 가능한 프로그램 모듈이라고 정의를 하였다. 그리고 컴포넌트가 갖추어야 할 조건을 다음과 같이 정의하였다.

- Pluggability
- Reusability
- Interface oriented Usage
- 컴포넌트 사용자에게는 Black box 형태 제공
- 컴포넌트 개발자에게는 White box 형태 제공
- Framework 기반 (UI 컴포넌트 제외)

컴포넌트 정의 외에도 주목해야 할 사항은 하나의 컴포넌트로 정의할 범위는 어떻게 잡을 것이라는 문제와 Persistent 데이터 영역을 어떻게 컴포넌트에 포함시킬 것인가 하는 문제가 존재하였다. 컴포넌트 범위를 너무 작게 잡으면, 이를 구현 시 시스템 관리상 문제뿐만 아니라 성능 저하를 가져올 위험을 가지고 있고, 너무 크게 잡으면, 컴포넌트 자체 재사용성에 제약이 커질뿐만 아니라 구현 시 이것 역시 시스템에 부담을 줄 소지가 있다. 또한, 컴포넌트가 패키징 및 Pluggable하게 하기 위해서는 Persistent 데이터 영역이 반드시 컴포넌트에 같이 포함되어야 하는데, 컴포넌트 범위가 너무 작게 잡힐 경우는 데이터 모델링상에 문제가 발생하고 데이터 무결성 등의 사항에 문제를 일으키게 된다.

본 프로젝트에서는 엔터프라이즈 포탈의 하나의 모듈을 대상으로 하는 System Boundary를 하나의 컴포넌트 범위로 잡았다. 그 안에 속한 하위 System Boundary는 모두 이 컴포넌트 내부 구



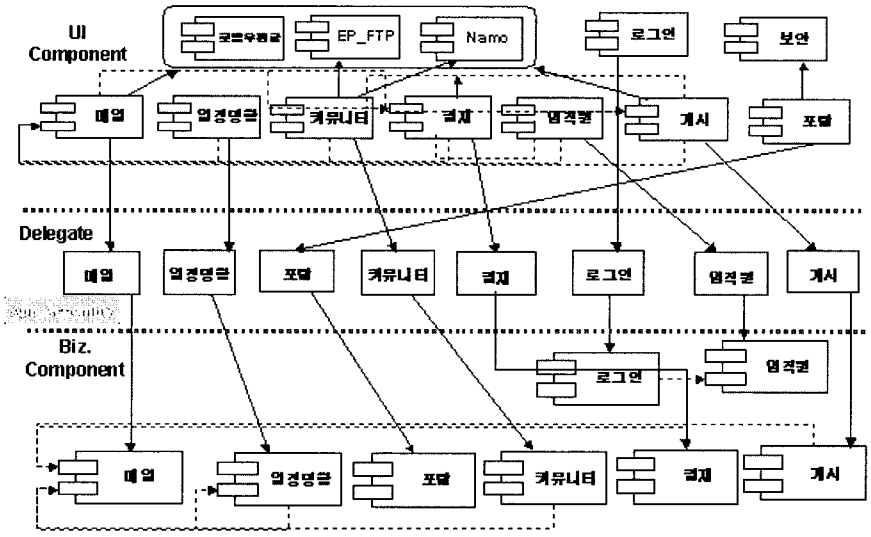
(그림 8) Component 단위

성요소로 포함이 되며, 이 System Boundary들이 가진 인터페이스는 이 컴포넌트에서 제공하는 인터페이스 종류로 포함되게 설계하였다. 이에 대한 개요를 나타낸 그림은 (그림 8)과 같다.

그리고, 각 하위 System Boundary의 Persistent 데이터는 모두 통합하여 전체 컴포넌트 단위 하나당 하나로 데이터 모델링을 하였다.

3.5. 컴포넌트 종류

본 엔터프라이즈 포탈 시스템에서 컴포넌트 유형을 크게 UI 컴포넌트, 각종 컴포넌트로부터의 서비스를 연계하여주고 조합해주는 Assembly 컴



(그림 9) 모듈간 연동

포넌트, 그리고 서비스를 제공하는 Business 컴포넌트로 구분하여 설계하였다.
전체 시스템에서 각 컴포넌트간의 연동되는 형태는 (그림 9)와 같다.
Delegate[3,4,5]부분은 웹 계층에 설치되어 각종 컴포넌트가 제공하는 인터페이스를 가지고 웹 계층에 서비스를 제공해주는 역할을 한다. 각 Delegate는 컴포넌트의 인터페이스 하나당 하나의 Delegate를 갖도록 설계하였다.

3.6 컴포넌트 활용

하나의 시스템은 컴포넌트들과 이를 조합하여 사용하는 어플리케이션 코드로 이루어진다. 시스템 개발 생산성을 높이기 위해서는 어떻게 제공되는 컴포넌트들을 잘 활용하여 구성할 수 있어야 하는데 이는 컴포넌트간을 어떻게 쉽게 잘 조립하여 서비스를 제공할 수 있느냐에 달려 있다 하겠다.
본 프로젝트에서 개발된 시스템은 세 가지의 연동(조립) 포인트를 가지고 있다.
첫째는 타 모듈과 화면상의 연동을 가능하게 하는 부분이다. 각 모듈의 Front controller[4] 가 이 역할을 담당하며, 해당 모듈의 Front controller에 요청을 재전송하여 처리를 의뢰하는 방식이다.
둘째는, 자신의 모듈이 아닌 다른 모듈의 컴포넌트로부터 서비스를 제공받아야 할 경우 이를 담당하는 연동부분이다. 설계상에서 Business Delegate가 이 역할을 수행한다. 각 Delegate에서 표준 인터페이스를 통해 서비스를 제공하므로 여러 개의 컴포넌트로부터 서비스를 받아 이를 조합하여 화면에 보여 주는 것도 가능하게 된다.
셋째는, 컴포넌트간의 연동 부분이다. 각 컴포넌트는 서비스를 제공하기 위한 여러 개의 인터페이스를 제공하고 있는데, 각 인터페이스는 Session Facade [4]를 통해 구현되어 있다. 컴포넌트간의 정보나 서비스 이용이 필요할 경우, 이 인터페이스를 이용하여 제공 받게 된다.

각 어플리케이션 개발자들은 컴포넌트 내부 내용을 알 필요 없이 관련 컴포넌트에서 제공하는 인터페이스 정의서를 가지고 개발을 진행하면 되므로 각 모듈별로 병렬적으로 개발을 진행할 수가 있다.

4. 효과 및 향후 과제

4.1 효과

원래 CBD 방법론이 의도했던 대로 생산성 향상을 기대하려면 재사용 가능한 컴포넌트가 일부 존재해야 한다. 하지만, 본 프로젝트에서는 재사용 가능한 컴포넌트가 없었던 바, 시스템을 설계하면서 각 모듈별로 재사용 가능하면서 독립적인 부분을 컴포넌트화 설계를 진행하였다.
CBD를 적용하여 본 프로젝트를 진행하여 얻을 수 있었던 대표적인 효과는 다음과 같다.
첫째, 초기에 컴포넌트화된 설계를 진행하면서 너무 세분화된 설계로 인해 코드가 더 복잡해지는 상황이 발생하지 않은가 하는 우려가 있었지만, 실제 결과는 오히려 프로그램 본 수가 감소한 것을 볼 수 있었다. 〈표 3〉은 이전 버전의 엔터프라이즈 포탈 시스템과 CBD 방법론을 적용하여 새로이 개발한 엔터프라이즈 포탈 시스템이 프로그램 본 수를 비교한 것이다. 여기서 CBD 방법론을 개발된 시스템은 이전 버전보다 기능이 더 추가됨에 불구하고 본 수가 줄어든 것에 주목하자.

〈표 3〉 프로그램 본수 비교: CBD 적용 vs CBD 미적용

항 목	CBD 미적용 EP - 일반 설계	CBD 적용 EP
적용 기술	JS P, Servlet	JSP, Servlet, JavaBeans,EJB, J2EE 디자인 패턴
Web Tier 본 수	1976	1410
App. Tier 본 수	572	912
전체 본 수	2548	2322

둘째, 컴포넌트화된 설계로 각 비즈니스 기능들이 모듈화됨에 따라, 관련된 중복 코드가 없어지고, 한 컴포넌트에서 비즈니스 기능이 집중 관리됨으로써 전문화된 서비스를 제공하고 사용할 수 있게 되었다. LDAP 관련 기능에 대해서 이전 버전에서는 각 모듈별로 필요기능을 각각 구현을 하는 형태이었으나 이것이 컴포넌트화되면서 타 모듈에서는 해당 컴포넌트로부터 서비스를 받기만 하는 형태가 되었다. 따라서 이전 버전의 LDAP 사용률에 비하여 80%정도 감소를 가져왔고, 에러 발생도 대폭 감소시키는 효과를 얻을 수 있었다.
셋째, 새로운 조합된 서비스 제공을 위한 개발 생산성이 대폭 향상되었다. 비즈니스 기능들이 컴포넌트화되어 이를 이용하는 인터페이스들이 제공됨에 따라, 화면 단에 대한 독립성을 보장하는 형태가 되었다. 이로 인해, 웹 브라우저를 통해서 서비스를 제공하는 것 외에도 다양한 클라이언트에 대한 서비스도 자유자재로 가능하게 되었다. 따라서, 모바일 서비스의 경우, 기존의 비즈니스 컴포넌트를 그대로 사용하게 됨에 따라, 상당한 개발 생산성 효과를 가져올 수 있었다.

4.2 향후 과제

CBD 방법론을 적용하려면 개발 조직의 수준 및 역량, 보유하고 있는 재사용 리소스 그리고 개발하려는 시스템의 특성을 고려하여 이에 맞게 적용되어야 한다.
본 프로젝트에서는 개발 조직은 객체지향 개발에 익숙한 상태이었고, 각종 디자인 패턴을 사용할 수 있는 상태였다. 보유하고 있는 재사용 리소스는 없는 상황이라 시스템 설계를 진행하면서 필요 부분은 컴포넌트 설계를 같이 진행해야 하는 상황이었다. 개발 대상은 엔터프라이즈 포탈 시스템이고, 이에 대한 각 서비스들에 대한 영역이 비교적 명확하여 비교적 컴포넌트 범위를 정하기에 용이한 형태였다.

본 시스템에서 앞으로 개선해야할 CBD에 관련된 향후 과제를 살펴보면 다음과 같다.

첫째, 구현 면에서 개발된 컴포넌트들의 독립성이 강화되어야 한다.
본 프로젝트의 컴포넌트는 설계 측면에서 재사용성 및 사용 측면에서의 편리성을 만족시키고는 있으나, 컴포넌트 배치 관련 사항은 미비한 사항이 많다. 컴포넌트 자체 변경이 일어났을 경우, 이를 이용하는 클라이언트에 대한 영향을 최소화하면서 컴포넌트 재배치를 할 수 있는 개선이 필요하다.
둘째, 컴포넌트 패키징 및 배포 방안 수립이 필요하다.
엔터프라이즈 포탈 시스템과 연동되어 운영되는 타 시스템이 많은 바, 경우에 따라서는 서비스를 제공하기 위해서 인터페이스를 제공하거나, 클라이언트가 되는 시스템에게 컴포넌트를 배포해야 하는 경우가 발생하게 된다. 이를 위한 체계적인 컴포넌트 패키징 및 배포 방안 수립을 하여야 하겠다.
셋째, 구현된 컴포넌트들은 프로젝트 진행 여건상 특정 벤더에 맞추어 개발된 것이 대부분이다. 따라서, 컴포넌트 자체를 배치 환경에 보다 구애 받지 않고 사용하기 위해서는 다양한 벤더를 지원할 수 있는 형태로 보완이 되어 재사용성을 높여야 하겠다. 또한, 개발된 컴포넌트를 체계적으로 유지보수 및 배포가 가능하게 하기 위해서는 재사용 Repository 구축이 반드시 필요하다.

참고문헌

[1] 삼성SDS, 삼성SDS Innovator CBD 방법론 version 1.0, 삼성SDS, 2002

[2] 삼성SDS, 솔루션 개발 품질매뉴얼 version 1.2, 삼성SDS, 2002

[3] Floyd Marinescu, EJB Design Patterns WILEY, 2002

[4] Deeppak Alur, John Crupi, Dan Malks, Core J2EE Design Patterns , Prentice Hall, 2001

[5] 이재익,김형준,김희영,안승규,안성화,김인로, Java 웹 성능 향상기법 , 삼성SDS 멀티캠퍼스, 2002년

저자 약력

이 경 배

1983년 동국대학교 전자계산학과(경영학사)

1997년 연세대학고 전자계산전공(공학석사)

1982년 삼성생명 정보시스템실

1995년 삼성전자 전략기획실

2000년 건국대학교 경영대학 (겸임교수)

2000년 삼성SDS 정보기술연구소 (연구소장)

2003년 삼성SDS 정보전략실 (상무)

자격 1993년 12월 정보처리기술사

저서

1999. 1 클라이언트/서버와 인트라넷
(이경배외 3인 공저, 시스템통합연구소 발행)

1999. 3 핵심 정보기술 총서(1,2,3권)
(이경배외 21인 공저, 한울아카데미 발행)

2000.11 포스트 지놈 시대의 생물정보학
(이경배외 7인 역저, 한울아카데미 발행)

이 메 일 : richard.lee@samsung.com

박 준 성

1978년 서울대학교 경영학과(학사)

1984년 서울대학교 경영학과(경영학석사)

1988년 미국 Ohio State University 전산학 및 시스템공학 박사

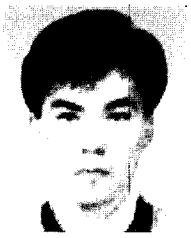
1987~1989년 미국 Louisiana 주립대 조교수

1989~2000년 미국 University of Iowa 부교수

1998~2000년 미국 OR/경영과학회(INFORMS)산하 통신 시스템연구회 회장

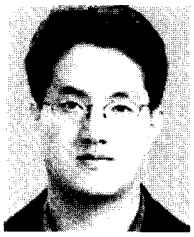
2001-현재 삼성SDS 상무, CTO 및 CKO (겸)첨단소프트웨어공학센터장

이 메 일 : june.park@samsung.com



이 재 익

1992년 한양대학교 정밀기계공학과 (공학사)
1994년 한양대 교 정밀기계공학과 자동제어전공 (공학석사)
1994년 - 1997년 삼성전자 근무
1998년 - 현재 삼성 SDS 근무
2002년 - 현재 삼성 SDS 사내교수
관심분야: CBD, Web Services, J2EE, EAI, Enterprise Design Patterns, Enterprise Architecture
이 메 일 : jakelee@samsung.com



강 금 석

1996 서울대학교 산업공학과(학사)
1998 서울대학교 산업공학과(석사)
1998-현재 삼성SDS 정보기술연구소 책임연구원
관심분야 : CBD, SOA, Architecture, 개발방법론
이 메 일 : keumseok.kang@samsung.com