CBD 와 EJB 기반의 뱅킹 시스템 설계 및 구현

정승재° 김 수 동
숭 실 대 학 교 컴 퓨 터
sjjung@selab.soongsil.ac.kr

Design and Implementation of Banking System based on CBD and EJB

Seung-Jae Jung° Soo-Dong Kim
Dept. of Computer Science, Soongsil University

요 약

소프트웨어의 경제성, 시장 경쟁력 확보를 위한 소프트웨어의 재사용은 소프트웨어 공학의 주요 이슈가 되고 있다. 그 중 컴포넌트와 컴포넌트 기반의 소프트웨어 개발은 재사용성을 확보할 수 있는 가장 주목 받는 방안으로 제시되고 있으며 많은 기법이나 지침들이 제안되고 있다. 기존의 캡슐화와 정보은폐를 핵심 개념으로 하는 OOD(Object Oriented Development)에 의한 소프트웨어 개발 방법은 이미 재사용성, 유지보수성, 무결성, 안정성 등의 많은 장점으로 인해 학계와 산업계에서 이미 많이 보편화 되어 있고 널리 이용되어 지고 있다. 하지만 CBD(Component Based Development)와 EJB(Enterprise JavaBeans)에 기반하여 컴포넌트 어플리케이션을 개발하는데 있어서는 그 적용 사례가 드물고 활요성 또한 검증된 바가 거의 없다. 따라서 본 논문에서는 대형 분산 시스템이라 할 수 있는 뱅킹 시스템에 CBD, EJB, UML(Unified Modeling Language)을 적용해 봄으로써 소프트웨어 개발시의 그 실무적인 유용성을 검증해 본다.

1. 서론

최근 몇 년 동안에 소프트웨어 개발이 급속도로 복잡해져 왔다. 클라이언트/서버 형태, 다양한 플랫폼의 지원, 그리고 보다 복잡하고 다양한 사용자 인터페이스를 지원하는 어플리케이션의 요구 사항으로 인해 개발자들은 어플리케이션 개발에 있어서 새로운 접근을 시도하고 있다. 컴포넌트와 컴포넌트 기반의 소프트웨어의 개발은 재사용성을 확보할 수 있는 가장 주목 받는 방안으로 제시되고 있으며 많은 기법이나 지침들이 제안되고 있다. 기존의 캡슐화와 정보은폐를 핵심 개념으로 하는 OOD에 의한 소프트웨어 개발 방법은 이미 재사용성, 유지보수성, 무결성, 안정성 등의 많은 장점으로 인해 학계와 산업계에서 많이 보편화 되어 있지만, CBD와 EJB에 기반하여 컴포넌트 어플리케이션을 개발하는데 있어서는 그 적용 사례가 드물고 활요성 또한 검증된 바가 거의 없다. 따라서 본 논문에서는 대형 분산 시스템이라 할 수 있는 뱅킹 시스템에 CBD, EJB, UML을 적용해 봄으로써 소프트웨어 개발시의 그 실무적인 유용성을 검증해 본다.

본 논문의 2장에서는 관련 연구로서 CBD, EJB, 뱅킹시스템에 대해서 설명하고, 3장에서는 뱅킹 시스템의 컴포넌트 식별 과정과 함께 컴포넌트의 인터페이스와 워크플로우를 기술한다. 4장에서는 EJB 컴포넌트 플랫폼을 고려한 상세 설계를 설명하고, 5장에서는 컴포넌트 구현에 대해 설명한다. 마지막으로 6장에서는 결론을 내리고자 한다.

2. 관련연구

2.1 CBD

컴포넌트 개발 방법론(CBD)으로는 Sterling 사의 CBD96 표준과 ICON Computing 사의 Catalysis, HP 의 Fusion, Compuware 사의 UNIFACE 등이 발표되고 있으나 아직 학계나 업계에서도 연구 중인 단계이다. 이들 컴포넌트 개발 방법론이 기존의 OMT 등과 같은 객체지향 방법론과 가장 다른 점은 컴포넌트 추출 과정의 요구라고 할 수 있다. 제안되고 있는 대부분의 컴포넌트 개발 방법론들은 객체지향 모델링을 기본으로 하며, UML 을 기본 모델링 도구로 사용하고 있다. 이들 방법론들은 상위 수준의 Use Case 나 개념 수준의 클래스들로부터 후보 컴포넌트를 추출하도록 제안하고 있으며 동일한 Use Case 나 클래스가 서로 다른 컴포넌트에 동시에 할당되는 경우를 배제하고 있다.

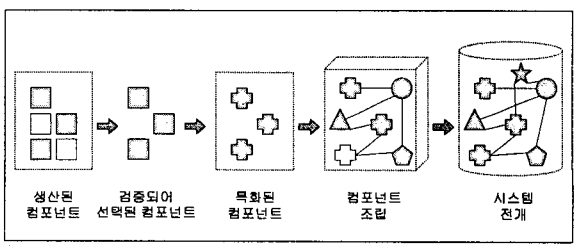

그림 1. 컴포넌트 기반의 개발 4단계

2.2 EJB

EJB는 다중 계층(Multi-Tier)의 분산형 객체지향 자바 어플리케이션을 개발하고 보급하기 위한 컴포넌트 아키텍처로서 선(Sun)사에서 개발하였다. EJB는 확장성 있는 어플리케이션 서버 컴포넌트들을 지원하는 여러 서비스들을 제공함으로서 비즈니스 어플리케이션들을 컴포넌트 단위로 쉽게 작성할 수 있도록 한다. EJB는 트랜잭션 처리 모니터, 웹 서버, 데이터베이스 서버, 어플리케이션 서버 등과 같은 트랜잭션 처리 시스템에서 운영될 수 있으며, EJB 컴포넌트 모델은 서버 컴포넌트들을 지원하기 위해 자바빈 컴포넌트 모델을 확장한 것이다. 서버 컴포넌트란 어플리케이션 서버에서 실행되는 어플리케이션 컴포넌트를 의미하는데, 이는 어플리케이션을 개발하기 위해 다른 컴포넌트들과 결합될 수 있다.

2.3 Banking System

대형 분산 시스템이라 할 수 있는 뱅킹 시스템은 네트워크 성능(Network Performance)에 민감하고, 분산 환경에서의 트랜잭션(Transaction), 무결성(Integrity)이 보장되어야 하며, Scalable, 신뢰적인(Reliable) 시스템의 성격을 지닌다. 뱅킹 시스템의 은행 업무에 대해 간단히 살펴보면 다음과 같다. 고객 관리(Customer Management)는 개인이나 기업이 은행 거래를 개시하고자 하는 경우 실명을 확인한 후 해당 고객에 대한 정보를 등록, 조회, 수정, 취소 하는 업무이다. 계좌관리(Account Management)는 은행에서 다루는 금융 상품을 관리하고 신규 계좌 개설과 기존 계좌의 해지를 담당한다. 그리고 계좌 원장과 고객별 계좌 목록 등의 조회 업무도 포함된다. 수신관리(Deposit Management)는 개인이나 기업이 그들의 자금을 은행에 저축 또는 출납대행 등의 목적으로 예금을 하거나 필요한 때 언제든지 환급을 받을 수 있으며 은행이 그 자금을 합리적으로 운용하여 얻어진 이익의 일부를 이자로 지급 받을 수 있는 업무이다.

3. 컴포넌트 식별 단계

이 단계에서는 이미 산출된 Use Case 모델과 객체 모델을 이용하여 Use Case/Class 메트릭(Matrix)을 작성한후, 클러스터링을 통하여 후보 컴포넌트을 식별해 낸다. 식별된 컴포넌트에 대해서 컴포넌트 다이어 그램을 작성하고 컴포넌트 인터페이스와 컴포넌트 워크플로우를 기술한다.

3.1 Use Case/Class 메트릭

FOCUS(컴포넌트 방법론)에서 제안하는 클러스터링 알고리즘을 이용하여 공통 컴포넌트를 식별하기 위한 Use Case/Class 매트릭를 작성한다. 먼저 공통의 Use Case와 Class들을 대상으로 Use Case/Class 맵핑 테이블을 만드는데 처음 가로줄은 Class, 가장 왼쪽 세로줄은 Use Case로 둔다. 각 Use Case별로 사용하는 클래스들을 생성,삭제,수정,읽기의 관계로 표시하는데 C(Create), D(Delete), W(Write), R(Read)을 이용해 테이블의 내용을 채워 넣는다. 그리고 클러스터링 알고리즘을 이용하여 Use Case와 Class가 생성, 수정, 삭제의 관계인 경우 이를 하나로 모으는데 C, D, W 관계가 서로 겹치는 직사각형의 클러스터가 추출되는데 이것이 컴포넌트의 기본 단위가 될 수 있으며, 경우에 따라서는 하나 이상의 클러스터가 더 모아져서 컴포넌트가 될 수도 있다.

	Use Cse Name	Cus. Journal	Customer	Private	Corporation	Group	Customer_TX
SM1	Teller확인						
SM2	고객 저널 등록	C					
CM1	고객 등록		C	C	C	C	C
CM2	고객 수정		W	W	W	W	C
CM3	고객 삭제		W	W	W	W	C
CM4	고객 조회		R	R	R	R	

표 1. Use Case/Class 메트릭

3.2 컴포넌트 식별과 컴포넌트 다이어그램

추출되고 정제된 컴포넌트들을 대상으로 컴포넌트 다이어그램을 생성하는데, 이는 UML의 컴포넌트 다이어그램을 확장한 형태로서 컴포넌트 내부에 객체 모델이 표현된다.

<고객관리>

고객 정보를 관리한다. 은행의 고객 정보를 등록, 조회, 수정 삭제한다. 고객 관리 업무와 관련된 Journaling 을 담당한다.

<수신관리>

수신계좌와 관련된 계좌 개설 및 해지, 입금/출금/이체를 관리한다. 수신 계좌와 관련된 Journaling과 Bookkeeping을 관리한다.

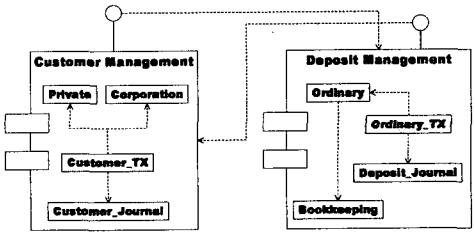

그림 2. 컴포넌트 다이어 그램

3.3 컴포넌트 인터페이스와 컴포넌트 워크플로우 정의

분석 단계에서의 Use Case 모델과 객체 모델을 검토하여 비즈니스 기능을 추출하는데 이를 위해서 각 컴포넌트에 대한 Use Case와 클래스 다이어 그램을 그룹화하여 후보 컴포넌트의 인터페이스 기능과 다른 컴포넌트 내부의 클래스와의 관계를 식별한다. 추출된 컴포넌트 인터페이스 명세서에는 인터페이스 이름, 공용 기능의 이름과 설명, 파라미터 리스트, 반환값의 타입등이 기술된다. 그리고 컴포넌트 인터페이스 각 공용 기능에 대한 순차도를 작성하여 인터페이스와 컴포넌트안의 객체 사이에 관계를 보여준다. 한 예로, 신규 계좌개설(openAccount)에 대한 컴포넌트 워크플로우를 정의한 부분을 보면 다음과 같다.

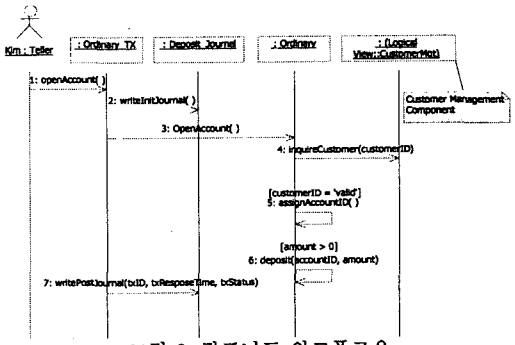

그림 3. 컴포넌트 워크플로우

4. 컴포넌트 플랫폼을 고려한 상세 설계 단계

컴포넌트의 구현은 컴포넌트가 저장, 관리되는 플랫폼에 따라 많이 달라지게 되는데 본 논문에서는 EJB 플랫폼 환경에서의 상세 설계 단계를 기술하고자 한다. 먼저 객체 모델에서의 클래스를 EJB 플랫폼의 컴포넌트 빈으로 매핑하기 위한 과정이 필요하고 엔터티 빈에 대한 지속성 정의, 트랜잭션 정의가 요구된다. 엔터티 빈을 위한 지속성 테이블을 작성할 때는 빈 마다 데이터베이스 스키마 설계하고 기본 키도 정의한다.

4.1 세션 빈과 엔터티 빈의 정의

EJB 플랫폼에서의 빈의 종류에는 상태유지(Stateful) 세션 빈, 무상태(Stateless) 세션 빈, 엔터티(Entity) 빈이 있는데 객체 모델에서의 클래스들을 빈으로 매핑하기 위해서는 각각의 클래스가 지속성 데이터, 비즈니스 로직, 공유되는 상태, 참조하는 외부

데이터 등을 포함하는 여부에 따라 서로 다른 유형의 빈으로 매핑 된다. FOCUS에서의 빈 매핑 작업를 위한 지침 테이블(Instruction Table)을 보면 다음과 같다.

	General Abstract Class		Concrete Class					Mapping Types
					Business Logic			
	Interface	Abstract Class	Persistent Data	Transient Data		Shared State	Referencing External Data	
I1	T							Interface
I2		T						Abstract Class
I3			F	X	T	F	X	Stateless Session Bean
I4			F	T	T	T	X	Stateful Session Bean
I5			T	X	F			Entity Bean
I7			T	T	T	F	T	Entity Bean & Stateless Session Bean
I8			T	T	T	T	T	Entity Bean & Stateful Session Bean

표 2. 지침 테이블

객체 모델에서의 각 클래스 특징에 맞는 빈 매핑 타입이 위의 지침 테이블에 8가지로 정의되어 있다. 이를 이용해서 개발자는 다시 클래스와 빈을 최종적으로 매핑 시켜 놓은 빈 매핑 테이블(Bean Mapping Table)을 작성한다.

Class	Mapping Type	Name	Instruction #
Customer	AC	Customer	2
CustomerTX	SL	CustomerTXBean	3
Private	ET	PrivateCusBean	5
Bookkeeping	ET	BookkeepingBean	5

표 3. 빈 매핑 테이블

4.2 지속성 정의

엔티티 빈의 경우, 자료의 지속성을 보장하는 방법에는 빈 관리(Bean-management)와 컨테이너 관리(Container-management)로 나뉠 수 있는데, 빈 관리의 경우, 빈 개발자가 직접 소스상에서 데이터베이스를 접근하여 자료의 지속성을 보장하는 방법이고, 컨테이너 관리의 경우, 컨테이너에서 우선 키 클래스를 이용하여 자동적으로 관리해 줌으로서, 빈 개발자가 별도의 지속성 관리를 위한 코딩을 해 주지 않아도 된다. 두 가지 방법들은 각각 장단점을 가지고 있으므로 업무의 성격 상, 결정해 주어야 한다. 지속성 테이블에서 자료의 지속성의 성격을 정의한다.

4.3 트랜잭션 정의

EJB에서 트랜잭션을 관리하는 방법은 빈 관리 트랜잭션(Bean-managed transaction)과 컨테이너 관리 트랜잭션(Container-managed transaction)으로 나뉠 수 있는데, 세션 빈은 두 가지 방법 모두 가능하지만, 엔티티 빈의 경우는 컨테이너 관리 트랜잭션만이 가능하다. 트랜잭션이 필요하지 않는 경우, 전개 설명서(Deployment Descriptor)에 있는 트랜잭션 속성자를 'NotSupported' 라고 해 줌으로서 처리할 수 있다. 트랜잭션 테이블에서는 빈의 특성을 분석하여 트랜잭션 타입을 기술해 준다. 빈 트랜잭션 속성 테이블에서는 컨테이너 관리 트랜잭션인 경우 빈의 각 함수를 분석하여 기능별로 Isolation수준과 트랜잭션 속성을 기술한다.

Bean Name	Op. Name	Isolation Level	TX attribute
PrivateCusBean	registerCustomer()	SERIALIZABLE	TX_REQUIRED
	modifyCustomer()	SERIALIZABLE	TX_REQUIRED
	removeCustomer()	SERIALIZABLE	TX_REQUIRED

표 4. Operation Transaction Table

5. 구현(Implementation) 단계

EJB 기반의 컴포넌트 플랫폼에서 상세 설계 단계에서의 결과물과 EJB 컴포넌트 명세서를 기반으로 실제 컴포넌트 빈을 구현한다. 빈 클래스와 함께 홈/원격 인터페이스, 기본 키 클래스, 도우미 클래스도 함께 구현된다. 컴포넌트별 빈에 대해서 전개 설명서(Deployment Descriptor)를 작성하고 데이터베이스 스키마를 구축하며, 최종적으로 EJB 서버에 컴포넌트 빈을 Deploy 하기 위해 관련 클래스들을 JAR 파일로 묶어준다. 전개 설명서에는 빈 클래스, 홈/원격 인터페이스, 기본 키 클래스 이름이 기술되고 데이터베이스 연결에 관련된 설정과 기타 환경 설정에 관련된 내용이 포함된다. 그리고 구현된 빈에 대해서 빈의 완성도를 높이고, 에러를 찾기 위한 테스트 단계를 수행한다.

6. 결론

본 논문에서는 대형 분산 시스템(뱅킹 시스템)을 사례 연구 도메인으로 하여 컴포넌트에 기반한 소프트웨어 개발의 설계, 구현 과정을 구체적으로 제시하였다. 이 과정에서 우리는 CBD, EJB 를 적용한 컴포넌트 소프트웨어 개발의 실무적인 유용성과 활용성을 검증할 수 있었다. 실무적인 유용성의 측면에서 보면, 이러한 소프트웨어 개발의 가장 큰 장점은 소프트웨어의 재사용성 확보라고 생각한다. 소프트웨어의 재사용성 확보는 개발 기간의 단축, 개발 비용의 절약, 생산성 향상, 위험요소 축소, 향상된 일관성이라는 장점들로 확대된다. 또한 전체 프로젝트에서 복잡도를 감소시키고 대량의 병렬 개발을 지원하며, 시스템의 적응력(Flexibility)을 향상시키며, 점진적인 시험을 가능하게 하며 유지보수를 쉽게 한다는 장점을 가진다.

이러한 상용 컴포넌트를 이용하여 어플리케이션을 구축하기 위해서는 수집과 선택부터, 커스터마이즈, 조립 등의 기존의 방법론과는 차별화된 전략들이 요구되는데, 향후 연구 과제로 컴포넌트 기반의 어플리케이션을 구축하는 보다 정제된 CBD 방법론을 연구하고 있다.

참고문헌

[1] Short K., Component Based Development and Object Modeling, Sterling Software, 1997.

[2] Kozaczynski Wojtek and Booch G.,"Component-Based Software Engineering," *IEEE Software*, pp.34-36, Sept./Oct. 1998.

[3] Sun Microsystems, *Enterprise JavaBeans Specification*, at URL: http://www.javasoft.com, 1999.

[4] 유영란, 김수동, "객체지향 객체 모델의 컴포넌트 모델 전환 지침", 한국정보처리학회 '제 13 회 춘계 학술 발표논문집', 2000

[5] 유영란, 김수동, "UML 기반의 컴포넌트 인터페이스 추출 기법", 한국정보과학회 '99 추계학술대회', 1999

[6] 김철진, 김수동, "컴포넌트 일반성 향상 기법", '제 13 회 춘계 학술 발표논문집', 2000

[14] 이종국, 김수동, "효율적인 컴포넌트 설계 기법", '제 13 회 춘계 학술 발표논문집', 2000

[15] 김철진, 김수동, "컴포넌트 일반성 향상 기법", '제 13 회 춘계 학술 발표논문집', 2000

[16] 김수동, "실무자를 위한 소프트웨어 공학", 에드텍, 1999.