韓國 컴퓨터情報學會 論文誌
第11卷 第3號, 2006. 7.

2006-11-3-1-21

비 컴포넌트 자바 프로그램에서 EJB 프로그램으로의 변환 프로세스

이 성 은*

A Process for Transforming Non-component Java Programs into EJB Programs

Sung-Eun Lee *

요 약

본 논문에서는 기존의 비컴포넌트 자바 프로그램을 EJB 컴포넌트 프로그램으로 변환하기 위한 프로세스를 제안한다. 자바 프로그램의 재사용성을 높이기 위해 다음과 같은 방법으로 접근한다. 먼저 기존에 존재하는 비 컴포넌트 자바 프로그램으로부터 컴포넌트 모델에 적합한 구성 요소를 추출하며, 추출된 구성 요소를 중심으로 컴포넌트로 변환하기 위한 기법을 제시한다. 기존 프로그램 코드의 재사용을 극대화하며, EJB의 특성을 고려한 컴포넌트 구성이 가능하도록 클래스 클러스터링과 메소드 단위 중심의 클래스 재구성 방법을 혼합하여 제안한다.

Abstract

In this paper, we suggest a process that transforms non-component Java programs into EJB component programs. We approach following methods to increase reusability of existing Java-based programs. We extract proper factors from existing non-component Java programs to construct for component model, and we suggest a transformation technique using extracted factors. Extracted factors are transformed into EJB components. With consideration for reusability of existing programs and EJB's characteristic, we suggest a process that mixes class clustering and method oriented class restructuring.

▸ Keyword : 컴포넌트(Component), EJB(Enterprise Java Beans), 재공학(Re-engineering),

---

• 제1저자 : 이성은
• 접수일 : 2006.06.01, 심사일 : 2006.06.18, 심사완료일 : 2006.07.18
* 동서울대학 컴퓨터정보과 부교수
※ 본 논문은 동서울대학 산업기술연구소 지원으로 수행되었음

I. 서 론

가최근 소프트웨어 프로그램 개발의 특징은 그 규모가 커지고 빠른 속도의 개발을 요구하고 있어, 효과적인 소프트웨어 개발 방법과 기존 소프트웨어의 효율적인 재사용, 그리고 유지 보수가 절실하게 요구된다. 기존에 개발된 소프트웨어의 대부분 규모가 방대하고 복잡하여 새로운 시스템으로의 전환이 어려우며 합리적이고 단순한 소프트웨어 활용과 이를 위해서는 전반적인 소프트웨어의 설계나 원시코드의 생성 등에서 유지 보수를 위한 소프트웨어의 이해는 물론, 소프트웨어의 재사용성을 좀더 광범위하게 높여 주는 일 등이 필요하다[1][2][3][4].

본 논문은 이미 존재하는 자바(Java) 기반의 프로그램을 대상으로 하며, 기존의 비컴포넌트 형태로 개발된 프로그램에 재공학을 적용하여 적합한 컴포넌트 요소를 추출하고 이를 자바 기반 컴포넌트로 변환함으로써, 프로그램의 재사용성을 높여주는 컴포넌트 기반의 프로그램으로 변환시켜주는 방법에 대한 연구이다. 연구는 기존 자바 프로그램 중 서버 영역에 해당하는 프로그램을 EJB 컴포넌트로 변환하기 위한 프로세스와 컴포넌트 추출에 관한 방법을 제안한다.

기존의 컴포넌트 기반 소프트웨어개발 방법론들은 대부분 순공학 중심의 프로세스 위주이고 재공학 또는 역공학 측면에서는 상세한 가이드가 부족한 실정이다[5][6]. 따라서 본 논문은 코드 수준의 재공학적인 측면에서 기존 프로그램을 컴포넌트 기반 프로그램으로 변환시킬 수 있도록, 재공학 측면에서의 컴포넌트 변환 프로세스를 제안한다

II. 관련 연구

2.1 MESSA

컴포넌트 추출 과정에 초점을 두고 있으며, 컴포넌트 식별을 레가시 소프트웨어의 원시코드만을 대상으로 하고 있으며, 비즈니스 로직을 가지는 컴포넌트 추출의 한계를 가지고 있다. 원시코드를 클래스 중심의 리팩토링 기법으로 적용했기 때문에 추출된 컴포넌트를 변환하기 위한 지침이 부족하다[7].

2.2. 기존 역공학 프로세스

대부분의 역공학 과정은 개발된 원시코드를 초기 입력으로 자동화 도구와 일련의 역공학 기술과 프로세스를 통해 최상위 추상화 수준의 명세를 추출하는 데 초점을 두고 있다. 즉, 지원하는 역공학 도구 대부분이 원시코드를 분석하는 Bottom-Up 방식이므로 비즈니스 로직 보다는 코드 분석에 중심을 두고 있다[8][9]. 대표적인 방법론으로 UIRich System Redevelopment Methodology(USRM)과 Mission Oriented Architecture Legacy Evolution(MORALE)가 있는데, 각 방법론들은 역공학을 통하여 레가시 시스템의 자원을 추출가능하다는 전제하에 구성되어 있으며, 역공학의 세부 단계는 다루지 않고 있으며, 세부 단계의 활동과 산출물이 명확히 정의되어 있지 않아 개발 생명 주기의 전 단계를 지원하지 못한다[10].

2.3. 프로세스 지향 유지보수 모델

Boehm, Yau & Collofello, Chapin등에 의해 정리된 프로세스 지향 모델(Process Oriented Model)은 실시된 활동과 활동들의 실시 순서 측면에서 재공학을 위한 유지보수 프로세스를 설명하고 있다. 이 모델에서Hansen은 육하원칙을 결정하기 위해 노력하였으며, 결국 프로세스 지향 모델은 지식과 기술을 공유하고 과정을 이해할 수 있도록 하여 유지보수에 대한 잠재력을 증진시키는 결과를 가져왔다[11].

프로세스 모델링에 대한 확장된 연구는 SEI(Software Engineering Institute)가 주도했으며, SEI와 카네기멜론 대학의 DoD 소프트웨어공학 연구소는 프로세스 모델링과 CMM(Capability Maturity Model)을 통한 개선에 적극적이었다.

2.4. 조직 지향 유지보수 모델

Pressman, Gamalel-Din, Osterweil등에 의해 정리된 조직 지향 모델(Organizational Oriented Model)은 실시된 활동과 활동들 사이에서의 정보 흐름에 중심을 둔 모델이다. 이 모델은 유지보수 과정의 여러 단계에 관련된 조직이나 사람들, 그리고 그들 사이의 의사 소통과 관련된 채널을 보여준다[11]. 조직 지향 모델 프로세스 성숙 레벨(Process Maturity Level)을 적용할 수 있다.

2.5. ISO/IEC 12207

ISO/IEC 12207은 소프트웨어의 생명주기 동안 수행될 수 있는 활동을 다섯 개의 기본 프로세스와 여덟 개의 지원 프로세스 그리고 네 개의 조직 프로세스로 구분한다. 각 생명주기 프로세스는 여러 개의 활동으로 구성되고, 각 활동은 더 세분화된 여러 개의 작업으로 구성된다.

기본 생명주기 프로세스 중 개발 프로세스는 개발자가

수행할 활동을 정의하고 있으며, 유지보수 프로세스는 소프트웨어 제품의 유지보수 서비스를 제공하는 유지보수자 또는 유지보수조직이 수행할 활동을 정의한다. 즉 소프트웨어 제품을 현 상태로 유지하며 적합하게 운영되도록 변경관리를 하는 것을 의미하는데, 이 프로세스는 소프트웨어 제품의 전환 및 폐기까지 포함한다[12].

III. EJB 컴포넌트 변환 프로세스

본 논문의 컴포넌트의 변환은 기존 자바프로그램 환경에서 서버영역에 해당하는 프로그램을 대상으로 하며, 본 논문에서 제시하는 변환 단계는 다음 그림1과 같다.

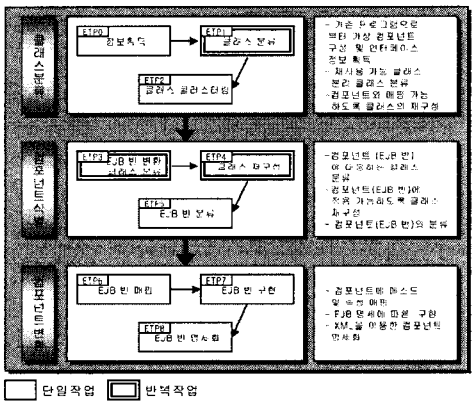


그림 1. EJB 프로그램 변환 프로세스
Fig. 1 EJB Program Transform Process

3.1. 정보획득 단계 (ETP0)

본 논문의 범위가 기존 프로그램의 재사용을 통하여 컴포넌트 기반의 프로그램을 생성하는 것 이므로, EJB 변환 기법을 진행하기 앞서 적용 프로그램에서 변환해야 할 컴포넌트에 관한 기본적인 범위를 식별할 필요성이 있다[13][14][15][16]. 컴포넌트의 식별 방법은 이미 기존에 연구된 CBSD 방법론들에 정의되어 있으며[17], 컴포넌트 식별 단계나 컴포넌트 인터페이스 정의 단계 후에 필요한 자료가 획득된다. 그림 2에 식별된 가상의 컴포넌트 예를 나타내었다. 이 단계에서 식별되는 컴포넌트는 기존의 비컴포넌트 프로그램으로부터 컴포넌트로 될수 있는 클래스들을 묶어 놓은 가상의 컴포넌트라고 할 수 있다.

가상 컴포넌트가 식별되면 컴포넌트 명세를 통해서 필요한 컴포넌트에 대한 정보를 표현한다. 컴포넌트 명세는 다음과 같은 내용으로 구성된다.

- 컴포넌트 ID
- 컴포넌트 명
- 키워드
- 구조설명
- 흐름도
- 참조 컴포넌트

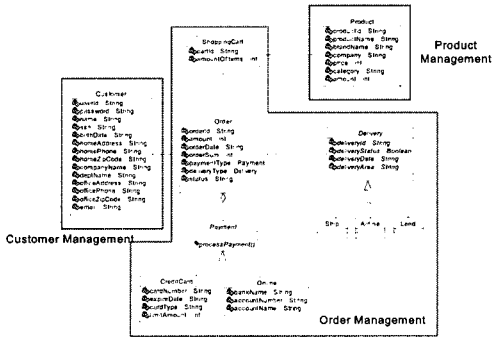


그림 2. 식별된 컴포넌트의 예
Fig. 2 An Example of Identified Components

컴포넌트가 식별되고 추출되면 각 컴포넌트 별로 인터페이스에 대한 명세가 준비되어야 한다. 컴포넌트 인터페이스는 자세하게 명세화 되어야 하는데, 본 논문에서 참조 사용한 컴포넌트 인터페이스의 내용은 다음과 같다[17].

- 인터페이스/메소드의 이름
- 기능 설명
- 파라미터 리스트
- 반환 값의 타입 지정

추출된 가상 컴포넌트의 규모(granularity)는 클래스(객체)와 같거나 여러 클래스가 모여서 하나의 컴포넌트를 형성할 수도 있다. 따라서 컴포넌트가 클래스와 같은 경우 클래스를 빈에 매핑 시켜 주어야 하며, 여러 개의 클래스로 이루어진 경우 빈을 하나 이상의 클래스로 매핑해 주는 과정이 필요하다.

3.2. 클래스 분류 (ETP1)

이 단계에서는 클래스들의 집합으로 이루어진 컴포넌트 내에서 재사용 가능한 클래스들과 분해 될 클래스들을 분류한 다음, 엔터프라이즈 빈에 매핑될 클래스들을 선택하여 메소드 단위로 분해한다. 클래스 분류 단계의 세부 과정은 그림3과 같다.

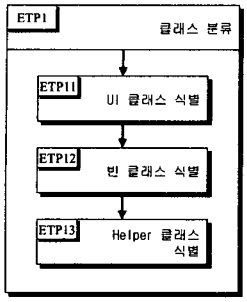


그림 3. 클래스 분류 세부 단계
Fig. 3 Detailed Phases of Class Classification

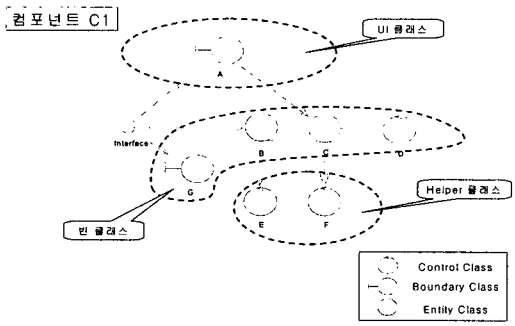


그림 4. 분류된 클래스들
Fig. 4 Classified Classes

3.2.1. UI 클래스(재사용 클래스) (ETP11)

사용자 인터페이스를 위한 메소드와 속성들로만 구성되어있는 클래스로서 UI 기능을 그대로 활용할 수 있는 경우 재사용이 가능하다. 재사용의 필요성이 많이 요구되는 경우 재사용성을 높이기 위해 UI부분을 자바 빈으로 컴포넌트화 하는 것이 가능하다.

3.2.2. 빈 대상 클래스(변환 가능한 클래스) (ETP12)

EJB의 엔터프라이즈 빈과 매핑될 메소드와 속성들로 구성되어있는 클래스로서 빈의 성격에 맞도록 재 구성되는 클래스이며, 다음과 같은 특성을 가진다.

- 데이터베이스 접근이 이루어지는 클래스
- 비즈니스 로직과 트랜잭션이 이루어지는 클래스
- 컴포넌트 식별 프로세스에서 정의된 컴포넌트 인터페이스와 직접 상호작용을 하는 클래스
- UI 클래스와 직접 상호작용을 하는 클래스
- 비즈니스 로직을 포함하는 UI 기능을 가진 클래스
- 빈 클래스로의 관계를 이루는 클래스

3.2.3. Helper 클래스(재사용 가능한 클래스) (ETP13)

빈 클래스를 도와주는 클래스로서 UI 클래스와 빈 클래스를 제외한 빈 클래스에서의 일방적인 관계를 이루는 클래스나 또는 Helper 클래스와 관계를 이루는 클래스를 말한다. 예외 처리를 위한 클래스도 이 클래스로 분류한다.

그림 4는 ETP1 단계에서 분류될 클래스들을 그룹으로 나누어 표현한 내용을 그림으로 나타낸 것이다.

3.3. 클래스 클러스터링(ETP2)

식별된 빈 대상 클래스들을 빈으로 변환하는 과정은 빈과 클래스사이의 매핑관계에 따라 달라질 수 있는데, 추출된 가상의 컴포넌트의 구조에 따라서 몇가지 경우의 매핑관계로 나누어진다.

컴포넌트의 구조에 따른 빈과 클래스간의 매핑 관계를 살펴보면

- 1 클래스 → n빈 (n = 1 또는 2)
- n 클래스 → 1빈 (n 〉 1)

  의 유형으로 구분된다.

위의 경우에서 둘째 경우에 해당하는 여러 개의 클래스로 이루어진 컴포넌트를 빈으로 변환하는 방법은 각각의 클래스단위로 빈에 매핑하는 방법과, 여러 개의 클래스로 묶어서 빈에 매핑하는 방법이 있다. 전자의 경우 생성된 빈이 세분화되어 활용되는 범위가 다양하다는 잇점이 있으나, 너무 작은 단위로 이루어진 빈들로 인한 성능의 효율성을 떨어뜨리고 개발시 필요한 노력이 더 소모된다는 단점이있다. 후자의 경우는 빈의 사용이 용이하며 재사용 범위가 적당한 크기를 유지할 수 있지만, 클래스들을 클러스터링하는 과정이 필요하다는 단점이 있다.

본 논문에서는 클래스들 간의 관계를 고려하여 클러스터링함으로써 관련있는 클래스들을 묶어 빈에 매핑시키는 방법을 적용한다.

클러스터링에는 [18][19]의 객체 클러스터링 방법을 적용한다.

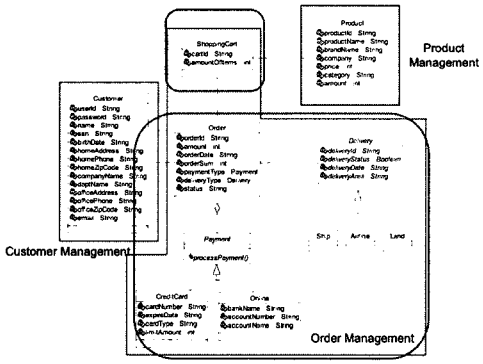


그림 5. 클러스터링된 빈 대상 클래스
Fig. 5 Clustering Classes for Bean

그림 2에 나타난 가상 컴포넌트 중에서 클래스 클러스터링을 통해서 빈으로 변환 가능한 클래스들을 그림 5에 둥근 사각형으로 표시하였다. 그림에서 여러 개의 클래스들로 형성된 Order Management 컴포넌트는 2개의 빈으로 변환하게 한다. 상속의 관계를 하나의 빈으로 구성할 경우 선택되어지는 하위클래스는 빈의 파라메터를 이용하여 구현하도록 한다.

3.4. 빈 변환 클래스 분류 (ETP3)

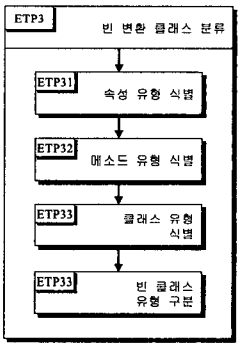


그림 6. 빈 변환 클래스 분류 단계
Fig. 6 Detailed Phases of Class Classification for Transform into Bean

이전 단계에서 구분한 재사용 클래스, 즉 UI 클래스와 helper 클래스들 이외에 빈으로 변환이 필요한 빈 클래스들에 대하여 빈과의 대응관계를 확인함으로써 클래스 분류 작업을 하는 단계이다. 클래스들을 빈으로 변환하기 위해 클래스와 빈과의 대응관계를 찾는 데에는 빈의 속성과 메소드의 특성을 고려하여, 빈의 시각으로 클래스들을 분류할 필요가 있다.

클래스가 가지고 있는 속성과 메소드의 특성을 살펴보면 빈의 관점에서 클래스의 유형을 나눌 수 있다.

3.4.1. 속성 유형 식별 (ETP31)

속성은 하나의 클래스에 의해 생성되는 여러 객체들에 의해 지속되는 데이터 값이라고 할 수 있다. 빈의 시각에서 속성을 고려한다면, 속성은 세션동안만 유지하는 일시적인 보존 상태의 속성과 영구적인 보존을 하는 속성으로 구분할 수 있다.

일시 보존 속성은 유지되는 관점에 따라 다음과 같이 두 가지로 분류할 수 있다.

non-conversational 속성 : 빈을 사용하는 모든 클라이언트에 대해서 어떠한 상태도 유지할 필요가 없는 속성으로, 주로 final과 static으로 선언된 속성과 임시 변수와 같은 속성이 여기에 속한다.

conversational 속성 : 클라이언트와의 지속적인 대화를 나타내는 속성으로, 이것은 빈을 호출하는 클라이언트와 관련된 상태를 유지해 주는 속성이다. 거의 대부분의 속성들이 이 부류에 속한다.

영구보존 속성은 객체를 실체화 하면서 자신의 속성을 지니게 되며 이러한 속성을 장기적인 목적을 위해서 데이터베이스나 파일 시스템 등을 이용하여 보조 기억 장치에 보존된다. 객체가 없어져도 기억된 속성들은 다시 복구되어 사용될 수 있다.

다음과 같이 속성 유형을 구분하여 사용한다.

- 속성 유형 구분
  1. An : 임시 속성
  2. Ac : 대화식 속성
  3. Ap : 영구 보존 속성

3.4.2. 메소드 유형 식별 (ETP32)

클래스를 빈의 관점으로 구분에 필요한 메소드의 유형은 홈 인터페이스 함수와 관련된 생성, 삭제, 찾기와 같은 데이터베이스의 명령과 밀접한 메소드와, 원격 인터페이스 및 빈 클래스에 정의되는 비즈니스 메소드와 연관되는 메소드의 부류로 나누어 진다. 다음과 같이 메소드 유형을 구분한다.

- 메소드 유형 구분
  1. Mq : 레코드 생성, 삭제, 찾기 등과 같이 빈의 홈 인터페이스 와 관련된 쿼리 메소드
  2. Mb : 원격 인터페이스의 비즈니스 메소드와 관련된 메소드

3.4.3. 클래스 유형 식별 (ETP33)

클래스의 유형은 ETP21, ETP22에서 분류한 속성과 메소드와 연관관계를 통해 분류할 수 있는데, [20]에서 정의한 빈의 관점에서 분류한 클래스와 객체의 속성과 메소드의 연관관계를 분석하여 식별할 수 있다. 클래스와 다음과 같이 속성과 메소드의 유형을 구분한다.

빈의 관점으로 분류할 수 있는 클래스는 먼저 다음과 같은 3가지 유형이 있다. 첫째 C1은 상태 없이 서비스를 제공하는 클래스로서, 속성을 가지지 않거나 메소드는 영속적인 자료와 관계없이 비즈니스 메소드만을 갖는 유형이다. 둘째 C2는 한 사용자의 여러 호출에서 유지되는 속성을 가지는 클래스이다. 셋째 C3는 여러 사용자간에 공유될 수 있는 비즈니스 객체를 표현하는 엔티티 클래스이다.[20]

3.4.4. 빈 클래스 유형 구분 (ETP34)

앞에서 살펴본 바와 같이 클래스는 빈으로의 대응이 가능한데, 클래스의 유형에 따라 빈과의 매핑 관계가 1 : 1 변환이 가능한 단일한 형태의 클래스인지 복합적인 형태의 클래스인지를 식별함으로써 빈 변환이 용이한 클래스를 선정할 수 있다. 변환될 빈의 유형과 그 클래스의 특성은 표1의 빈 매핑 테이블에 의해 분류가능하다.

3.5. 클래스 재구성 (ETP4)

컴포넌트 내의 빈으로 변환되어야 할 클래스는 빈과 단일 매핑이 가능한 클래스들을 앞 단계에서 분류해내었기 때문에, 빈과의 관계가 복합적인 유형의 클래스들에 대한 처리가 필요하다. 이러한 클래스들에 대한 처리는 클래스가 가지고 있는 메소드들을 기능적으로 분류함으로서 빈의 인터페이스 및 메소드로 배분한다.

표1. 빈 매핑 테이블
Table. 1 Bean Mapping Table

	속성			메소드							빈유형
				Mq			Mb				
	An	Ac	Ap	An	Ac	Ap	An	Ac	Ap	Not Ref.	
1	T	T	T	T	T	T	X	X	X	X	EB+SFB
2	T	T	T	T	F	T	X	T	X	X	EB+SFB
3	T	T	T	F	T	T	T	X	X	X	EB+SFB
4	T	T	T	F	F	T	T	T	X	X	EB+SFB
5	T	T	F	T	T	F	X	X	F	X	SFB
6	T	T	F	F	T	F	T	X	F	X	SFB
7	T	T	F	F	F	F	T	T	F	X	SFB
8	T	F	T	X	F	T	T	F	X	X	EB+SSB
9	T	F	T	T	F	T	F	F	X	T	EB+SFB
10	T	F	T	T	F	T	F	F	X	F	EB
11	T	F	F	F	F	F	T	F	F	X	SSB
12	F	T	T	F	T	T	F	X	X	X	EB+SFB
13	F	T	T	F	F	T	F	T	X	X	EB+SFB
14	F	T	F	F	T	F	F	X	F	X	SFB
15	F	T	F	F	F	F	F	T	F	X	SFB
16	F	F	T	F	F	T	F	F	X	T	EB+SSB
17	F	F	T	F	F	T	F	F	X	F	EB
18	F	F	F	F	F	F	F	F	F	X	SSB

X: don't care, EB:엔티티빈, SFB : 상태 세션빈, SSB : 무상태 세션빈

단일 매핑이 가능한 빈 클래스들을 제외한 나머지 클래스들을 메소드 기능 분류를 통해 분해한다. 본 논문에서는 메소드 종류를 다시 4가지로 나누어 그림 7과 같이 분류한다.

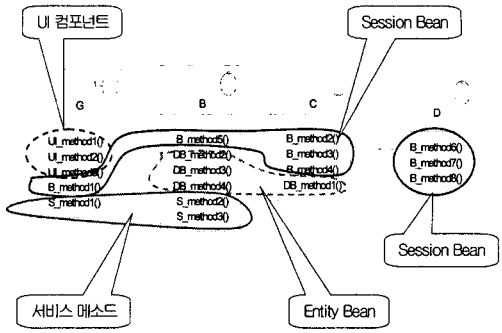


그림 7. 메소드 기능 분류에 의한 컴포넌트 분해
Fig. 7 Components Disassembly by Functional Classification of Methods

- UI 메소드(user interface method) : UI에 작용 또는 UI와 관련된 메소드
  예)ChattingClient() : 채팅방 화면의 UI 설정
  message() : 채팅창에 뿌려질 메시지 설정
- 데이터베이스 메소드(Database method) : DB의 사용에 관련된 메소드
  예) check() : 사용자와 패스워드를 체크한다.
- 서비스 메소드 (service method) : 어플리케이션 서버에서 제공되어지는 기본적인 서비스에 해당하는 메소드
  예) dbConnect() : DB에 연결한다.
  SetLog() : 접속자의 로그를 기록한다.
- 비즈니스 메소드 (business method) : 그밖에 일반적인 비즈니스 로직을 구현하는 메소드
  예) ReadMessage() : 메시지를 리턴한다.
  WriteMessage() : 메시지를 보낸다.

먼저 빈 클래스 중 UI기능을 가지는 클래스는 UI 메소드와 관련 속성들을 추출하여 새로운 UI클래스로 구성이 되고, 이 클래스는 그림 29의 UI 클래스에 속하게 된다. 이렇게 구성된 UI 클래스를 묶어서 하나의 UI 컴포넌트로 분해 되어지는데 이 때 컴포넌트 식별 프로세스에서 정의된 인터페이스와 컴포넌트 명세서를 참고하여 새로운 인터페이스를 정의하고 컴포넌트 명세서를 작성한다. 다음으로 두 종류의 빈을 구성하기 위해서 데이터베이스 메소드들을 추출하여 엔티티 빈을 구성하고 비즈니스 메소드들을 추출하여 세션 빈을 구성한다.

이렇게 구성된 빈은 표 2의 빈 매핑 테이블에 기록한다. 빈을 도와주는 Helper 클래스는 모두 묶어서 하나의 패키지를 만든다. 서비스 명세서는 service 메소드들 각각의 기능과 속성들의 기술을 작성하여 빈 배치자에게 제공한다.

표 2. 빈 매핑 테이블의 예
Table. 2 An Example of Mapping Table

컴포넌트 이름	빈 이름	메소드 시그네처	사용하는 메소드	사용하는 속성	메소드 기능
C1 컴포넌트	엔티티빈	DB_method1()	B_method2()		DB에 ... 추가한다.
		DB_method2()			DB에 ... 검색한다.
		DB_method3()	B_method6()	b	DB에 ... 검색한다.
		DB_method4()			DB에 ... 삭제한다.
	세션빈-1	B_method1()			... 기능을 수행한다.
		B_method2()	DB_method1() B_method6()	e	... 기능을 수행한다.
		B_method3()		f	... 기능을 수행한다.
		B_method4()	DB_method1()	a	... 기능을 수행한다.
		B_method5()	DB_method4()	b	... 기능을 수행한다.
	세션빈-2	B_method6()		d	... 기능을 수행한다
		B_method7()	B_method3() B_method8()		... 기능을 수행한다
		B_method8()		g	... 기능을 수행한다

표 2에서 세션빈-2는 그림 7의 클래스 D에 해당하는 것으로6.3의 클래스 분류 단계에서 단일 매핑 관계로 빈으로 변환될 클래스가 세션빈으로 확정된 것이고, 어둡게 칠해진 엔티티 빈과 세션빈-1은 여러 개의 클래스들을 메소드 기능으로 분해하여 엔티티 빈과 세션 빈으로 재구성한 상태이다.

3.6. 빈 분류 (ETP5)

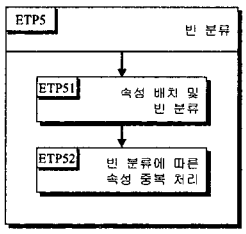


그림 8. 빈 분류 세부 단계
Fig. 8 Detailed Phases of Bean Classification

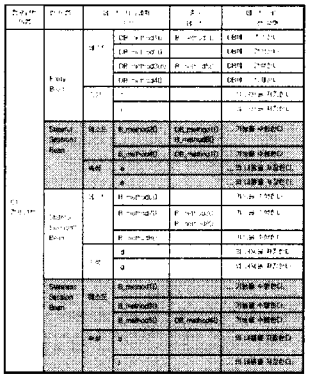
3.6.1. 속성 상태 분류에 의한 속성 배치 및 빈 분류 (ETP51)

메소드의 기능에 의해 생성된 빈에는 각 메소드들이 사용하는 속성을 배치 시켜야 한다. 세션 빈의 경우 그 속성의 상태에 따라 무상태 세션 빈과 상태 세션빈으로 다시 분해한다.

표 1의 어두운 부분에 해당되는 빈 중에 세션 빈부분은 다시 한번의 분류가 필요한데, 대화식 속성을 사용하는 메소드들은 상태 세션 빈으로 그 외에 임시 속성을 사용하는 메소드들과 속성을 사용하지 않는 메소드들은 무상태 세션 빈으로 구성한다.

이렇게 구성된 빈을 표 3과 같은 빈 분류 테이블에 기록한다.

표 3. 빈 분류 테이블
Table. 3 Bean Classification Table



임시 속성 : a,d,e,g
대화식 속성 : b,f

3.6.2. 빈 분류에 따른 속성의 중복 처리 (ETP52)

지금까지의 과정 가운데 같은 클래스에 포함되어있던 메소드들이 분리 되는 과정에서 각각 다른 빈에 소속되었을 때, 한 메소드가 사용하는 속성이 다른 메소드에서도 사용될 경우가 존재할 수 있다. 이런 경우, 같은 속성을 복제하여 각각의 빈에 위치시키며, 이러한 속성의 사용에는 자바의 동기화(synchronized) 메소드를 이용함으로써 자료의 불일치가 발생하지 않도록 한다.

3.7. 빈과의 매핑 (ETP6)

빈을 구성하기위한 빈 분류 테이블을 기준으로 빈을 기술한다. 빈의 기술은 빈의 사용자와 빈, 빈과 빈간의 상호작용을 위한 홈 인터페이스와 리모트 인터페이스를 정의하고 빈 명세서에 빈의 기능을 기술한다.

표 4는 표 3에 정의된 엔티티 빈의 메소드와 매핑될 홈 인터페이스와 의 관계를 표시한다.

표 4. 엔티티 빈의 인터페이스 매핑
Table. 4 Interface Mapping of Entity Bean

메소드 시그네처 /속성	메소드 기능 /속성 설명	홈 인터페이스 (엔티티 빈)
DB_method1()	DB에 추가한다.	create()
DB_method2()	DB에 검색한다.	find ()
DB_method3()	DB에 검색한다.	find ()
DB_method4()	DB에 삭제한다.	remove()

이밖에 빈 분류 테이블에 정의된 세션 빈의 메소드들은 리모트 인터페이스와 각각 매핑시킨다.

3.8. 빈 구현 (ETP7)

빈의 구현은 앞에서 추출된 빈들에 대하여 EJB 스펙을 따르는 다음 절들의 사항들을 고려하여 작성한다.

빈 구현 단계의 세부 사항을 그림 9에 나타 내었다.

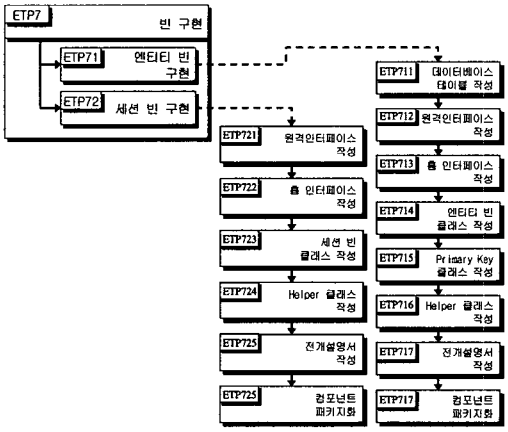


그림 9. 빈 구현 세부 단계
Fig. 9 Detailed Phases of Bean Implementation

3.8.1. 엔티티 빈 구현 (ETP71)

(1) 데이터베이스 테이블 작성 (ETP711)

빈이 다루는 데이터 베이스의 테이블을 생성한다.

(2) 원격 인터페이스 작성 (ETP712)

- 이름을 빈이름 + Remote로 정의한다.
- javax.ejb.EJBObject를 상속하고 클라이언트가 호출할 비즈니스 메소드를 정의한다.
- 리모트 인터페이스의 각각의 메소드는 엔티티 빈 클래스의 메소드와 대응해야 한다.
- 리모트 인터페이스의 메소드의 시그니처(signatures)는 엔티티 빈 클래스의 대응되는 메소드의 시그니처와 동일해야 한다.
- 전달인자와 반환 값은 반드시 유효한 Java RMI 형이어야 한다.
- throws절은 반드시 java.rmi.RemoteException을 포함해야 한다.

(3) 홈 인터페이스 작성 (ETP713)

홈 인터페이스는 클라이언트가 엔티티 빈 객체를 생성하고 그것을 찾도록 해 주는 메소드 들을 정의한다. 엔티티 빈 객체의 수명주기에 관련된 메소드들을 정의한다.

- 이름을 빈이름 + Home으로 정의한다.
- 다음의 create 메소드에 대한 요구사항을 만족해야 한다.
  - 대응되는 엔티티 빈 클래스에 위치한 ejbCreate 메소드와 동일한 수와 형태의 전달인자를 가져야 한다.
  - 엔티티 빈 객체의 리모트 인터페이스를 반환해야 한다.
  - throws절은 대응하는 ejbCreate와 ejbPostCreate 메소드의 throws절에 설정된 예외들을 포함해야 한다.
- 홈 인터페이스에 있는 모든 찾기 메소드는 엔티티 빈 클래스에 있는 찾기 메소드 와 대응된다.
- 홈 인터페이스에 정의된 찾기 메소드 의 이름은 find 로 시작하고, 엔티티 빈 클래스에 있는 메소드의 이름은 ejbFind 로 시작된다.
- 홈 인터페이스의 찾기 메소드 의 시그니처들(signatures)을 정의하는 규칙들은 다음과 같다.
- 전달인자의 개수와 형은 엔티티 빈 클래스에 있는 대응되는 메소드의 것들과 동일해야 한다.
- 반환값은 엔티티 빈의 리모트 인터페이스 형이거나 그런 형들의 집합(Enumeration type)이다.
- throws 절의 exception들은 엔티티 빈 클래스의 대응되는 메소드에 나타나 있는 것을 포함해야 한다.
- throws 절은 javax.ejb.FinderException과 javax.ejb.RemoteException을 담고 있다.

(4) 엔티티빈 클래스 작성(ETP714)

다음과 같은 요구사항을 만족시키도록 작성한다.

- 이름을 빈이름 + Bean으로 정의한다
- EntityBean Interface를 구현한다.
  - EntityBean interface는 Serializable interface를 상속(extends)한 Enterprise- Bean interface를 상속한다.
  - EntityBean interface는 ejbActivate나 ejbLoad와 같은 몇 개의 메소드를 선언하는데, 이것은 엔티티 빈 클래스에서 반드시 구현되어야 한다.
- 클래스는 public으로 선언한다.
- 클래스는 abstract나 final로 선언될 수 없다.
- 0개 이상의 ejbCreate와 ejbPostCreate method를 구현한다.
- 찾기 메소드를 구현한다. (bean-managed persistence 일 경우에만)
- 비즈니스 메소드를 구현한다.
- empty constructor를 가지고 있다.
- finalize 메소드를 구현하지 않는다.

(5) Primary Key 클래스 작성(ETP715)

프라이머리 키는 반드시 정의하여야 하는 것은 아니지만, 일반적으로 엔티티 빈에는 고유한 primary key 클래스를 따로 정의하여 사용하도록 한다. 이름을 빈이름 + Remote로 정의한다.

(6) 추가 helper 클래스 작성(ETP716)

빈의 처리와 관련하여 추가된 예외 처리용 클래스들을 추가로 작성한다.

(7) 전개 설명서(deployment discriptor)작성 (ETP717)

EJB 에서는 전개 설명서를 XML DTD로정의하고 있으며, 이 정의를 따라서 전개설명서를 작성한다.

(8) 컴포넌트 패키지화 (ETP718)
각 빈을 위한 하부 클래스 파일들로부터 하나의 JAR 파일을 생성한다. 컴포넌트를 이루는 빈들의 JAR 파일의 집합이 서버에 전개되어 하나의 컴포넌트로서 기능하게 한다.

3.8.2. 세션 빈의 구현 (ETP72)

엔티티 빈과는 달리 영속성을 갖는 데이터 베이스의 처리가 없다. helper 클래스 추가, 전개설명서 작성, 컴포넌트의 패키지화 등은 엔티티빈과 유사하게 적용한다.

(1) 원격 인터페이스 작성 (ETP721)
- 이름을 빈이름 + Remote로 정의한다.
- 엔티티 빈과 동일하게 javax.ejb.EJBObject를 상속하고 클라이언트가 호출할 비즈니스 메소드를 정의한다.
- 리모트 인터페이스의 각각의 메소드는 세션 빈 클래스의 메소드와 대응해야 한다.
- 리모트 인터페이스의 메소드의 시그니처는 세션 빈 클래스의 대응되는 메소드의 시그니처와 동일해야 한다.
- 전달인자와 반환값은 반드시 유효한 Java RMI 형이어야 한다.
- throws 절은 반드시 java.rmi.Remote Exception을 포함해야 한다.

(2) 홈 인터페이스 작성 (ETP722)
반드시 파라미터가 없는 create() 메소드를 하나 포함하여야 한다.
홈 인터페이스는 클라이언트가 세션 빈 객체를 생성하고 그것을 찾도록 해 주는 메소드들을 정의한다. 세션 빈 객체의 수명주기에 관련된 메소드들을 정의한다.
이름을 빈이름 + Home으로 정의한다.

(3) 세션 빈 클래스 작성(ETP723)
- 이름을 빈이름 + Bean으로 정의한다.
- SessionBean Interface를 구현한다.
- ejb 메소드(ejbActive(), ejbPassive(), cjbCreate(), ejbRemove())를 구현한다.

3.9. 빈 명세화 (ETP8)

구현된 빈을 이용하여 시스템을 개발하기위해서는 빈에 대한 명세를 가지고 있는 것이 유리하다. EJB 빈의 경우 전개서에 XML형태로 정보를 제공하고 있으나, 빈단위의 세부정보는 없다. 그러나 빈단위로 내부 정보를 제공한다면 개발 및 유지보수에 있어서 아주 효과적으로 사용할 수 있으므로, 본 논문에서는 XML을 이용하여 EJB 빈의 명세를 표현한다. 본 논문에서 제공하는 EJB 빈 명세에는 빈 이름, 빈 타입, 인터페이스 명, 속성 이름, 메소드 이름, 메소드 설명의 항목으로 명세하였지만 DTD의 확장성을 고려하면 추가적인 명세가 가능하다.

IV. 사례 연구 및 평가

위에서 제시한 빈 추출 기법을 바탕으로 채팅 기능을 갖는 BBS 사스템을 컴포넌트 기반의 프로그램으로 변환하기 위하여, EJB 빈으로 매핑하였다. 사전 단계에서 획득된 자료로는 컴포넌트 다이어그램과 컴포넌트 명세서, 컴포넌트 인터페이스 명세서 등이며, 컴포넌트 다이어그램은 그림 10에 표현하였다.

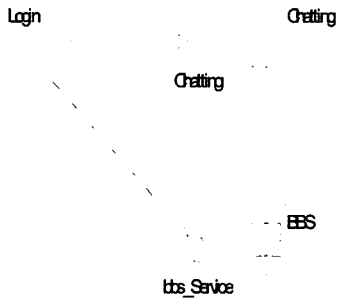


그림 10. BBS 시스템의 컴포넌트 다이어그램
Fig. 10 Component Diagram of BBS

그림 11처럼 실험 대상BBS 시스템은 login, chatting, bbs의 세개의 컴포넌트로 이루어져 있다.

실험 대상 BBS 시스템의 클래스 다이어그램은 그림 35와 같으며, 사전 단계에서 획득한 문서들을 적용하여 클래스 다이어그램 안의 클래스들을 그룹핑할 수 있다.

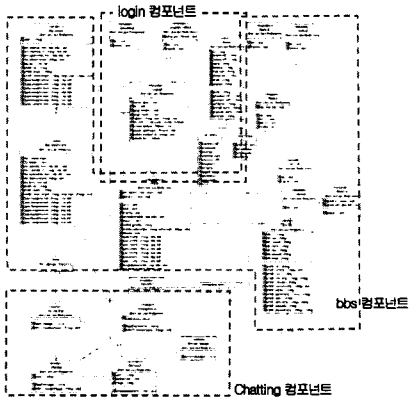


그림 11. BBS 시스템의 클래스 다이어그램
Fig. 11 Class Diagram of BBS

4.1. 클래스 분류

실험 대상이 되는 기존 BBS 시스템의 클래스 다이어그램으로부터 표 5와 같은 컴포넌트별로 클래스를 분리할 수 있다.

분리된 컴포넌트들은 그 구성을 분류하고자 하는 클래스들로 분류하여 표로 만들수 있다. 각 컴포넌트별 클래스 분류 테이블은 각각 표 6, 표 7, 표 8과 같다.

표 5. 클래스 분류
Table. 5 Class Classification

컴포넌트 이름	클래스 이름	인터페이스
Login	LoginUI, Person, FullMember, SemiMember GroupInfoUI, GroupInfoModifyUI, MessageBox	login(id:String, pass:String):void person_add(per:Person):void person_update(per:Person):void person_read(per_id:String):void person_list():Person
bbs	bbs_Service, bbs_Service_Impl, Server, bbsUI, ReplyUI, ModifyUI, ReadUI, WriteUI, SearchUI, ResultUI, bbs_Info, MessageBox	bbsAdd(tempInfo:bbs_Info):void bbsList():void bbs_read(bbs_id:String):bbs_Info Control_getInfo():String setInfoCount(tempid:String, tempCount:String):void bbsUpdate(tempInfo:bbs_Info):void bbsDelete(tempid:String):void query1(temp:String):bbs_Info query2(temp:String):bbs_Info query3(temp:String):bbs_Info query4(temp:String):bbs_Info query5(temp:String):bbs_Info query6(temp:String):bbs_Info query7(temp:String):bbs_Info
Chatting	Chatting, Chatting_Impl, Server, MessageVector, ReceiveMessage, ChattingUI, MessageBox	receiveMessage():Vector WriteMessage(msgstr:String):void

표 6. login 컴포넌트의 클래스 분류표
Table. 6 Class Classification Table of login Component

클래스 분류	클래스 명	클래스 설명
UI 클래스	GroupInfoUI	사용자 가입화면
	GroupInfoModifyUI	사용자 정보수정화면
	MessageBox	에러 메시지 출력화면
Bean 클래스	LoginUI	로그인 화면과 로그인 비즈니스 로직
helper 클래스	Person	사용자 정보의 필드

표 7. chatting 컴포넌트의 클래스 분류표
Table. 7 Class Classification Table of chatting Component

UI 클래스	ChattingUI	채팅의 메인화면
	MessageBox	에러 메시지 출력화면
Bean 클래스	ReceiveMessage	메시지를 받는 기능
	Chatting	채팅의 주 비즈니스 로직의 인터페이스
	Chatting_Impl	채팅의 주 비즈니스 로직
	Server	채팅의 비즈니스 로직의 호출을위한 서버(rmi 사용)
helper 클래스	MessageVector	메시지를 저장하는 공유 벡터

표 8. bbs 컴포넌트의클래스 분류표
Table. 8 Class Classification Table of bbs Component

클래스 분류	클래스 명	클래스 설명
UI 클래스	ReplyUI	게시판 답답화면
	ModifyUI	게시판 수정화면
	ReadUI	게시판 읽기화면
	WriteUI	게시판 쓰기화면
	SearchUI	게시판 검색화면
	ResultUt	게시판 결과화면
	MessageBox	에러 메시지 출력화면
Bean 클래스	bbsUI	게시판의 메인화면및 비즈니스 로직 호출
	bbs_Service	게시판의 주 비즈니스 로직의 인터페이스
	bbs_Service_Impl	게시판의 주 비즈니스 로직
	Server	게시판의 비즈니스 로직의 호출을위한 서버 (rmi 사용)
helper 클래스	bbs_Info	게시판의 정보

4.2. 클래스 클러스터링

구분된 가상의 컴포넌트내에 여러 개의 클래스가 포함된 경우 클래스 클러스터링을 통해 빈으로 변환될 클래스들의 범주를 정한다. 본 논문의 실험 대상 프로그램에서는 login 컴포넌트의 person 클래스와 FullMember, SemiMember 클래스의 상속관계가 존재하지만 person 클래스는 이미 helper 클래스로 분류되어 있으므로 이 예에서는 적용 대상이 없다.

4.3. 빈메소드 기능 분류에 의한 클래스 재구성

지금까지 구한 컴포넌트들의 클래스 분류표로부터 각각의 컴포넌트별로 자신에 속해있는 클래스들이 가지고 있는 메소드의 기능을 중심으로 분류된 표를 작성한다. 메소드의 기능은 UI 메소드, service 메소드, DB메소드, business 메소드로나누어 분류한다. 먼저bbs 컴포넌트에 속해있는 클래스들에 속한 메소드들의 기능별 분류는 표 9, 표 10, 표 11에 나타내었다.

표 9. bbs 컴포넌트의 메소드 기능별 분류(I)
Table. 9 Functional Classification(I) of bbs Component's Method

메소드종류	메소드 시그네쳐	소속 클래스	메소드기능설명
UI메소드	write():void	bbsUI	게시판 쓰기화면 호출한다.
	read():void	bbsUI	게시판 읽기화면 호출한다.
	search():void	bbsUI	게시판 검색화면 호출한다.
	Refresh():void	bbsUI	게시판을 새로고친다.
Service 메소드	DBConnct():void	Bbs_Service_Imple	DB(데이)베이스와 연결을 한다
		Server	원격 메소드의 rmi연결을 한다

표 10. bbs 컴포넌트의 메소드 기능별 분류(II)
Table. 10 Functional Classification(II) of bbs Component's Method

DB 메소드			
	Query1(searchWord:String):bbs_Info	bbs_Service_Imple	제목으로 게시물을 조회한다.
	Query2(searchWord:String):bbs_Info	bbs_Service_Imple	내용으로 게시물을 조회한다.
	Query3(searchWord:String):bbs_Info	bbs_Service_Imple	작성자로 게시물을 조회한다.
	Query4(searchWord:String):bbs_Info	bbs_Service_Imple	제목과 내용으로 게시물을 조회한다.
	Query5(searchWord:String):bbs_Info	bbs_Service_Imple	제목과 작성자로 게시물을 조회한다.
	Query6(searchWord:String):bbs_Info	bbs_Service_Imple	내용과 작성자로 게시물을 조회한다.
	Query7(searchWord:String):bbs_Info	bbs_Service_Imple	전체 게시물을 조회한다.
	bbs_add(bbs_item:bbs_Info):void	bbs_Service_Imple	게시물을 추가한다.
	bbs_read(bbs_id:String):bbs_Info	bbs_Service_Imple	게시물을 읽어온다.
	bbs_update(bbs_item:bbs_Info):void	bbs_Service_Imple	게시물을 수정한다.
	bbs_delete(tempid:String):void	bbs_Service_Imple	게시물을 삭제한다.
	bbs_list():bbs_Info	bbs_Service_Imple	게시물을 리스팅한다.
	getInfo_id():String	bbs_Service_Imple	게시물의 아이디를 얻어온다.
	getInfo_count():String	bbs_Service_Imple	게시물의 조회수를 얻어온다.
	setInfo_count(tempid:String, tempCount:String):void	bbs_Service_Imple	게시물의 조회수를 설정한다.

표 11. bbs 컴포넌트의 메소드 기능별 분류(III)
Table. 11 Functional Classification(III) of bbs Component's Method

Business 메소드	bbsAdd(temp_info:bbs_info) void	bbs	게시물을 추가한다
	bbsList() void	bbs	게시물을 리스트한다
	Control_getInfo() String	bbs	게시물의 개수(리스트)를 얻어온다
	setInfoCount(tempId:String, tempCount:String) void	bbs	게시물의 조회수를 설정한다
	query1(temp:String) bbs_info	bbs	제목으로 게시물을 조회한다
	query2(temp:String) bbs_info	bbs	내용으로 게시물을 조회한다
	query3(temp:String) bbs_info	bbs	작성자로 게시물을 조회한다
	query4(temp:String) bbs_info	bbs	제목과 내용으로 게시물을 조회한다
	query5(temp:String) bbs_info	bbs	제목과 작성자로 게시물을 조회한다
	query6(temp:String) bbs_info	bbs	내용과 작성자로 게시물을 조회한다
	query7(temp:String) bbs_info	bbs	전체 게시물을 조회한다
	bbsUpdate(temp:info:bbs_info) void	bbs	게시물을 수정한다
	bbsDelete(tempId:String) void	bbs	게시물을 삭제한다

위 표들에서 나타난 것과 같은 기능별 분류를 통해서 컴포넌트들을 빈 단위로 재 구성할 수 있는데, DB메소드들을 중심으로 엔티티 빈을, 비즈니스 메소드들을 중심으로 세션 빈을 구성할 수 있도록 한다. 이상과 같이 획득한 매핑 테이블은 1차 빈 매핑 테이블로서 표 12, 표 13과 같다.

login 컴포넌트 내의 메소드 기능 중심의 분류는 표 14와 같다.

표 14의 메소드 기능별 분류표로부터 표 15와 같은 빈 중심으로 메소드를 편성시킨 1차 빈 매핑 테이블을 얻을 수 있다.

표 12. bbs 컴포넌트의 빈 매핑 테이블(I)
Table. 12 Bean Mapping Table(I) of bbs Component

컴포넌트 이름	빈 이름	메소드 시그네처	사용하는 메소드	사용하는 속성	메소드 기능
bbs	Entity Bean	Query1(searchWord:String) bbs_info			제목으로 게시물을 조회한다
		Query2(searchWord:String) bbs_info			내용으로 게시물을 조회한다
		Query3(searchWord:String) bbs_info			작성자로 게시물을 조회한다
		Query4(searchWord:String) bbs_info			제목과 내용으로 게시물을 조회한다
		Query5(searchWord:String) bbs_info			제목과 작성자로 게시물을 조회한다
		Query6(searchWord:String) bbs_info			내용과 작성자로 게시물을 조회한다
		Query7(searchWord:String) bbs_info			전체 게시물을 조회한다
		bbs_update(bbs_info:bbs_info) void		userID String	게시물을 갱신한다
		bbs_updateInfo(tempId:String, bbs_info) void		user ID String	게시물의 조회수를 갱신한다
		bbs_delete(tempId:String) void		userID String	게시물을 삭제한다
		bbs_list() bbs_info			게시물을 리스트한다
		getInfo(tempId) String			게시물의 조회수를 얻어온다
		getInfo(countId:String) String			게시물의 조회수를 얻어온다
		setInfo(countId:String, tempCount:String) void			게시물의 조회수를 설정한다

표 13. bbs 컴포넌트의 빈 매핑 테이블(II)
Table. 13 Bean Mapping Table(II) of bbs Component

컴포넌트 이름	빈 이름	메소드 시그네처	사용하는 메소드	사용하는 속성	메소드 기능
bbs	Session Bean	bbsAdd(temp:info:bbs_info) void	bbs_add(bbs_item:bbs_info) void	bbs_services bbsService	게시물을 추가한다
		bbsList() void	bbs_list() bbs_info	bbs_services bbsService	게시물을 리스트한다
		Control_getInfo() String	getInfo_id() String	bbs_services bbsService	게시물의 개수를 얻어온다
		setInfoCount(tempId:String, tempCount:String) void	setInfo_count(tempId:String, temp_Count:String) void	bbs_services bbsService	게시물의 조회수를 설정한다
		query1(temp:String) bbs_info	Query1(searchWord:String) bbs_info	bbs_services bbsService	제목으로 게시물을 조회한다
		query2(temp:String) bbs_info	Query2(searchWord:String) bbs_info	bbs_services bbsService	내용으로 게시물을 조회한다
		query3(temp:String) bbs_info	Query3(searchWord:String) bbs_info	bbs_services bbsService	작성자로 게시물을 조회한다
		query4(temp:String) bbs_info	Query4(searchWord:String) bbs_info	bbs_services bbsService	제목과 내용으로 게시물을 조회한다
		query5(temp:String) bbs_info	Query5(searchWord:String) bbs_info	bbs_services bbsService	제목과 작성자로 게시물을 조회한다
		query6(temp:String) bbs_info	Query6(searchWord:String) bbs_info	bbs_services bbsService	내용과 작성자로 게시물을 조회한다
		query7(temp:String) bbs_info	Query7(searchWord:String) bbs_info	bbs_services bbsService	전체 게시물을 조회한다
		bbsUpdate(temp:info:bbs_info) void	bbs_update(bbs_item:bbs_info) void	bbs_services bbsService	게시물을 수정한다
		bbsDelete(tempId:String) void	bbs_delete(tempId:String) void	bbs_services bbsService	게시물을 삭제한다

표 14. login 컴포넌트의 메소드 기능별 분류
Table. 14 Functional Classification of login Component's Methods

메소드종류	메소드 시그네처	소속 클래스	메소드기능설명
UI메소드	combo_Box() void	LoginUI	콤보박스에 사용자 이름 설정한다.
	ok() void	LoginUI	OK버튼을 눌렀을때의 메소드로 호출한다.
	group_invoke() void	LoginUI	사용자 추가를 위한 UI를 호출한다.
	group_modify void	LoginUI	사용자 정보 변경을 위한 UI를 호출한다.
Transaction 메소드	check(id:String,pass:String) boolean	LoginUI	사용자의 아이디와 패스워드가 올바른지를 검사한다.
	person_add(per:Person) void	LoginUI	사용자를 추가한다.
	person_update(per:Person) void	LoginUI	사용자 정보를 변경한다.
	person_read(per_id:String) void	LoginUI	사용자 정보를 읽는다.
	person_list():Person	LoginUI	사용자 정보를 나열한다.
Business 메소드	login(id:String, pass:String) void	LoginUI	사용자를 인증한다.

chatting 컴포넌트의 메소드 기능 분류는 표 15와 같다.

표 15. login 컴포넌트의 빈 매핑 테이블
Table. 15 Bean Mapping Table of login Component

컴포넌트 이름	빈 이름	메소드 시그네처	사용하는 메소드	사용하는 속성	메소드 기능
Login	Session Bean	login(id:String, pass:String) void	check(id:String,pass:String) boolean		사용자를 인증한다.
	Entity Bean	check(id:String,pass:String) boolean			사용자의 아이디와 패스워드가 올바른지를 검사한다.
		person_add(per:Person) void			사용자를 추가한다.
		person_update(per:Person) void			사용자 정보를 변경한다.
		person_read(per_id:String) void			사용자 정보를 읽는다.
		person_list():Person			사용자 정보를 나열한다.

표 16. chatting 컴포넌트의 메소드 기능 분류
Table. 16 Functional Classification of chatting Component's Method

메소드종류	메소드 시그네처	소속 클래스	메소드기능설명
Business 메소드	ReadMessage():Vector	Chatting_Impl	메시지를 메시지 벡터로부터 읽어온다.
	WriteMessage(msg:string):void	Chatting_Impl	메시지를 메시지 벡터에 쓴다.
	receiveMessage():Vector	ReceiveMessage():Vector	일정시간 반복적으로 메시지를 읽어온다.
Service 메소드		Server	원격 메소드의 rmi관리를 한다.

표 16으로부터 재정리한 chatting 컴포넌트의 빈 매핑 테이블은 표 17과 같다.

표 17. chatting 컴포넌트의 빈 매핑 테이블
Table. 17 Bean Mapping Table of chatting Component

컴포넌트 이름	빈 이름	메소드 시그네처	사용하는 메소드	사용하는 속성	메소드 기능
Chatting	Session Bean	ReadMessage():Vector			메시지를 메시지 벡터로부터 읽어온다.
		WriteMessage(msg:string):void		userID	메시지를 메시지 벡터에 쓴다.
		receiveMessage():Vector	ReadMessage():Vector		일정시간 반복적으로 메시지를 읽어온다

4.4. 속성 상태분류에 의한 속성 배치 및 빈 분류

앞 절에서 분류한 빈 매핑 테이블에서 분류된 메소드들이 사용하는 속성에 대한 분류와 배치는 표 18, 표 19, 표 20, 표 21과 같은 속성상태 분류 표에 나타낸다.

표 18. bbs 컴포넌트의 속성상태 분류 표(I)
Table. 18 Attribute State Classification Table(I) of bbs Component

컴포넌트 이름	빈 이름	메소드 시그네처 /속성		호출하는 메소드	메소드 기능 /속성 설명
bbs	Entity Bean	메소드	Query1(searchWord:String) bbs_info		제목으로 게시물을 조회한다
			Query2(searchWord:String) bbs_info		내용으로 게시물을 조회한다
			Query3(searchWord:String) bbs_info		작성자로 게시물을 조회한다
			Query4(searchWord:String) bbs_info		제목과 내용으로 게시물을 조회한다
			Query5(searchWord:String) bbs_info		제목과 작성자로 게시물을 조회한다
			Query6(searchWord:String) bbs_info		내용과 작성자로 게시물을 조회한다
			Query7(searchWord:String) bbs_info		전체 게시물을 조회한다
			bbs_add(bbs_item:bbs_info, userID:String):void		게시물을 추가한다
			bbs_read(bbs_id:String):bbs_info		게시물을 읽어온다
			bbs_update(bbs_item:bbs_info, userID:String):void		게시물을 수정한다
			bbs_delete(temp_id:String, userID:String):void		게시물을 삭제한다
			bbs_list():bbs_info		게시물의 리스팅한다
			getinfo_id():String		게시물의 아이디를 얻어온다
			getinfo_count():String		게시물의 조회수를 얻어온다
			setinfo_count(temp_id:String, temp_Count:String):void		게시물의 조회수를 설정한다

표 19. bbs 컴포넌트의 속성상태 분류 표(II)
Table. 19 Attribute State Classification Table(II) of bbs Component

컴포넌트 이름	빈이름	메소드 시그네처 /속성		호출하는 메소드	메소드 기능 /속성 설명
bbs	Entity Bean	속성	Id : String		게시물 아이디
			Title : String		게시물 제목
			Content : String		게시물 내용
			Name : String		게시자 이름
			Date : String		게시 날짜
			Passwd : String		게시물 비밀번호
			Count : String		조회 회수

표 20. login 컴포넌트의 속성상태 분류 표
Table. 20 Attribute State Classification Table of login Component

컴포넌트 이름	빈 이름	메소드 시그네처 /속성		호출하는 메소드	메소드 기능 /속성 설명
login	Entity Bean	메소드	check(id:String,pass:Siring) :boolean		사용자의 아이디와 패스워드가 올바른지를 검사한다
			person_add((per:Person):void		사용자를 추가한다
			person_update(per:Person):void		사용자 정보를 변경한다.
			person_read(per_id:String):void		사용자 정보를 읽는다.
			person_list():Person		사용자 정보를 나열한다.
		속성	Id : String		사용자 아이디
			Person_idF:String		주민등록번호 앞
			Person_idP String		주민등록번호 뒤
			Name:String		사용자 이름
			Address:String		사용자 주소
			Mail_num:String		우편 번호
			Phon_num:String		전화 번호
			Email:String		메일 주소
			Passwd:String		비밀 번호
	Stateless Session Bean	메소드	login(id:String, pass:String):void	check(id:String,pass:String):boolean	사용자를 인증한다.

표 21. chatting 컴포넌트의 속성상태 분류 표
Table. 21 Attribute State Classification Table of chatting Component

컴포넌트 이름	빈 이름	메소드 시그네처 /속성		호출하는 메소드	메소드 기능 /속성 설명
chatting	Stateless Session Bean	메소드	ReadMessage() :Vector		메시지를 메시지 벡터로부터 읽어온다.
			receiveMessage() :Vector	ReadMessage() :Vector	일정시간 반복적으로 메시지를 읽어온다
	Stateful Session Bean	메소드	WriteMessage (msg:string):void		메시지를 메시지 벡터에 쓴다.
		속성	userID		사용자 아이디

4.5. 빈의 매핑과 구현

위에서 획득한 표들을 중심으로 재구성된 컴포넌트들을 3.9절에서 나타내었던 방법으로 빈의 메소드들을 매핑 시킨다.

매핑된 빈들은 3.10 절에 기술한 바와 같은 규칙을 적용하여 작성함으로써 구현한다. 엔티티 빈이 컨테이너-관리인지 아니면 빈-관리 인지는 기존 시스템의 환경, 사용 범위, 요구 사항 등을 고려하여 개발자의 판단에 의해 결정한다.

4.6. 평가

사례를 통해 본 논문에서 제시한 방법을 통해서 기존의 자바프로그램을 확보하고 있을 경우 컴포넌트를 추출하는 프로세스를 통해서 컴포넌트 기반 프로그램으로 변환하는 과정을 통해서 본 논문의 실험에서 이루어진 변환 대상 프로그램의 사용자 요구 단위 기능에 대한 결과와 추출된 컴포넌트 프로그램의 단위 기능의 결과가 동일한 기능성을 갖는 프로그램이라는 것을 확인할 수 있어 올바른 변환이 이루어진 것을 확인할 수 있다.

실험을 통해 확인한 본 논문의 제안 프로세스 중 클래스의 분류와 클러스터링을 통해서 클래스의 재사용성을 높일 수 있으며, 빈의 형성에 있어서 프로그램의 재사용과 EJB의 목적에 맞는 빈을 구성할 수 있다.

EJB 로의 변환에 있어서 본 논문의 제안 방법을 평가하기 위하여 실험 프로그램을 클래스와 빈을 1 : 1로 매핑하는 일반적인 방법과 비교해 봄으로써 본 논문의 특성을 평가하기위한 자료로 사용한다.

표 22. 제안 방법과 1:1 매핑 방법의 비교
Table. 22 The Comparison of Proposed Technique and 1:1 Mapping Technique

척도 \ 추출방법	T1	T2
생성 빈의 감소율	0.73	0.45
클래스 재사용률	0.14	0.00
평균 빈의 크기	1.50	1.00
평균 인터페이스 메소드의 수	6.17	5.75

T1:제안방법, T2: 1:1 매핑방법

비교내용의 항목내용과 그 분석결과는 다음과 같다.

- **생성 빈의 감소율**

$$\text{생성 빈의 감소율} = \frac{(\text{전체 클래스 수} - \text{빈의 개수})}{\text{전체 클래스 수}}$$

모든 클래스가 빈으로 변환되는 것을 가정할 때 실제로 생성된 빈의 개수를 뺀 나머지 만큼의 클래스가 줄어들기 때문에, 그 감소율을 측정할 수 있다.

실험의 결과로 1:1 방식에 비하여 생성되는 빈의 수가 현격하게 작기 때문에 감소율이 높다는 것을 확인할 수 있다.

- **클래스 재사용률**

$$\text{클래스 재사용률} = \frac{\text{재사용된 클래스의 개수}}{\text{전체 클래스 수}}$$

재사용된 클래스가 전체 클래스 가운데 차지하는 비율로서 값이 높을수록 재사용율이 좋은 것이다. 본 논문의 실험 결과를 살펴보면 재사용율이 높아진 것을 알 수 있다.

- **평균 빈의 크기**

$$\text{평균 빈의 크기} = \frac{\text{클래스 개수}}{\text{빈의 수}}$$

프로그램의 변환 과정에서 생긴 빈이 포함하고 있는 클래스의 수이며, 빈의 크기와 밀접한 관계가 있다.

- **평균 인터페이스 메소드 수**

$$\text{평균 인터페이스 메소드 수} = \frac{\text{인터페이스 메소드 수}}{\text{빈의 수}}$$

빈의 평균적인 크기와 복잡도와 연관이 있는 자료이며, 본 논문의 실험결과 제안 방법이 1:1 매핑 방법에 비하여 빈의 개수는 작은 반면 빈의 복잡도는 증가하는 결과를 가지고 있음을 알 수 있다.

비교 내용을 중심으로 본 논문에서 제안한 EJB 빈 변환 기법을 평가하기 위하여 표 22와 같은 결과를 확인할 수 있다.

표 23. 제안된 기법의 평가 척도
Table. 23 Evaluation Measurement of the Proposed Technique

평가 척도 \ 항목	CBSD 노력 (CBSD effort) 절감	모듈화 (Modularity) 증가	재사용성(Reusability) 증가		생산성(Productivity) 증가	
			클래스	빈	빈	스탑워어
컴포넌트 변환 프로세스 정의	↑	-	-	-	↑	↑
재사용 클래스 분석 (UI/Helper)	↑	↑	↑	-	↑	↑
클래스 클러스터링	↑	↑	↑	-	↑	↑
빈 클래스 분석 및 빈 매핑	↑	-	-	↑	↑	↑
클래스 재구성 (메소드정의)	↑	↓	-	↓	↑	↑
빈 완성화	↑	-	↑	↑	-	↑

↑ : 증가, ↓ : 감소, - : 해당 없음

표 23을 통해서 살펴 본 실험의 결과는 본 논문을 통해 제안된 기법을 적용하여 기존의 자바 프로그램에서 컴포넌트를 추출하는 과정에서 클래스를 메소드 중심으로 나누는 과정에서 클래스가 가지고 있는 고유의 모듈화 정도가 떨어지거나, 클래스 자체의 재사용성이 떨어지는 단점이 발생하기도 하지만, 전반적으로 S/W의 생산성의 향상과 CBSD 노력 부분의 감소를 가져올 수 있으며, 빈 추출시의 클래스 재사용과 생성된 빈의 재사용성의 증가를 가져올 수 있음을 알 수 있다.

V. 결론

본 논문은 순공학 측면의 컴포넌트 기반의 소프트웨어 개발 보다는 이미 보유하고 있는 기존 프로그램을 컴포넌트 기반의 프로그램으로 변환하는 재공학에 관한 연구이다.

본 논문에서는 기존의 자바 프로그램을 재사용성이 높은 컴포넌트 기반 프로그램으로 전환하기 위하여 컴포넌트를 추출하기위한 기법을 제시하였다. 특히 기존 프로그램의 재사용성을 극대화하기 위하여 직접적인 재사용이 가능한 클래스들을 수정 없이 최대한 사용하도록 제안하였으며, EJB 빈의 설계 및 구현 특성에 맞추어 관련 있는 메소드 중심으로 분할 통합하여 빈을 새롭게 생성시키도록 하였다.

본 논문에서 제시한 변환 프로세스를 채팅 기능을 포함하고 있는 BBS시스템에 적용함으로써 EJB 컴포넌트를 추출하는 과정을 보였으며, 이 실험을 통해 나타난 결과와 기존 프로그램의 클래스를 1 : 1방식으로 컴포넌트로 매핑하는 경우를 비교하여 본 논문에서 제안한 방법의 장단점을 고찰 하였다.

참고문헌

[1] Roger S. Pressman, Software Engineering A Practitioner's Approach, 5th Edition, McGraw-Hill, 2001.

[2] C. McClure, The Three R's of Software Automation, Prentice Hall.Inc, 1992.

[3] Ivar Jacobson, "Object-Oriented Software Engineering", Addison-Wesley, 1993.

[4] Jahannes Sametinger, "Software Engineering with Reusable Components", Spinger-Verlag, 1997.

[5] Desmon F.D'souza and A.C.Wills, Objects, Components, and Components with UML, Addison-Wesley, 1998.

[6] Wojtek Kozaczynski & G.Booch, "Component-basedSoftware Engineering," IEEE Software, Vol.15, No.5, pp.34~36, September/October, 1998.

[7] 이종호, "객체지향 코드로부터 컴포넌트 추출 프로세스 및 리팩토링 기법," 숭실대학교대학원 박사학위청구논문, 2002.

[8] Harry M. Sneed, "Program Interface Reengineering for Wrapping", Proceedings of the Working Conference on Reverse Engineering(WCRE) '97, 1997.

[9] Maozhen Li, Omer F. Rana, David W. Walker, "An XML-Based Component Model for Wrapping Legacy Codes as Java/CORBA Components", 1999.

[10] Gregory Abowd, et al, "MORALE-Missio Oriented Architectural Legacy Evolution," Proceedings International Conference on Software Maintenance'97, Bari, Italy, September 29-October 3, pp. 150-159, 1997.

[11] Thomas M. Pioski, Practical Software Maintenance, Wiley, pp.37~50, 1997

[12] ISO 12207, "ISO/IEC Standard for Information TechnologySoftware life cycle processes," ISO/IEC JTC/SC7, March, 1998.

[13] Sun Microsystems, Inc., Enterprise JavaBeans Specification, version 2.0, 2000.

[14] 양희석, 퍼펙트 EJB, 한빛미디어, 2002.

[15] 박천구, 문창수, EJB & WebLogic, 가메출판사, 2003

[16] Gilda Pour, "Java-Based Component Model for Enterprise Application Development", Technology of Object-Oriented Languages and Systems(TOOLS) USA '99 Conference, Santa Barbara, CA, Aug. 1999.

[17] 컴포넌트 생성 및 추출을 위한 기법 개발-II, 한국 전자 통신 연구원, 2000.

[18] 김동관, 김수동, "클라이언트 서버 및 분산 S/W 개발을 위한 객체 클러스터링 기법", 한국정보과학회논문지 25권 7호, 1998.

[19] 김철진, 조은숙, 김수동 "효율적인 객체지향 설계 및 성능 측정을 위한 정적/동적 메트릭", 한국정보과학회 논문지 25권11호, 1998.

[20] 박종성, 객체를 EJB(Enterprise Java Bean)기반 엔터프라이즈 빈으로의 매핑기법, 숭실대학교대학원 석사학위청구논문, 2000.

저 자 소 개

이 성 은

2001년 8월 : 숭실대학교 컴퓨터학과 공학박사

1993~ 현재 : 동서울대학 컴퓨터정보과 부교수