Journal of the Society of Korea Industrial and Systems Engineering
Vol. 31, No. 3, pp.01-07, September 2008.

EJB 2.0과 EJB 3.0의 소프트웨어 개발 생산성 비교 연구

이 명 호†

세명대학교 전자상거래학과

A Study on Comparison of Software Development Productivity with EJB 2.0 and EJB 3.0

Myeong-Ho Lee†

Department of e-Commerce, Semyung University

This paper proposes an object-oriented software development guidance and an evaluation index for the productivity related to EJB(Enterprise JavaBeans). EJB is a known successful standard model for LSDO(Large Size Distributed Object). However, there is no comparison research about the performance of EJB 2.0 and 3.0 with same identical platform. Quantitative analysis is supported as a part of LOC(Line Of Code) analysis. There is a limit to develop the updated software with no the specific evaluating index for the productivity of the software.

This work proposes an specific index for evaluating the productivity of new version EJB on a platform. Base on the result, the specific guidance of the developing software is obtained.

Keywords : Evaluation Index, Distributed Object, LOC, EJB 2.0, EJB 3.0

1. 서 론

디지털 컨버전스 시대에서의 운영환경 통합은 온-디맨드로 통합되며, 기반구조의 통합은 그리드나 유틸리티로, 개발통합은 통합개발 환경으로, 데이터베이스 통합은 데이터 허브나 EAI(Enterprise Application Integration)로, 사용자 인터페이스 통합은 X-인터넷으로, 마지막으로 프로세스 통합은 BPM이나 ERP로 통합화 및 표준화가 진화되고 있다[3]. 이러한 엔터프라이즈 환경에서는 이 기종 컴퓨터들 간에 프로그램을 분산시켜 부하를 줄여 시스템의 성능 저하와 네트워크 병목 현상을 줄일 수 있는 분산객체 구조가 필요하게 되었다. 또한 복잡한 시스템 요구조건을 신속히 구현하기를 원하는 유비쿼터스 정보화 시대에서는 다양한 객체 지향 및 분산 객체 개발 방법론을 거쳐 현재는 컴포넌트 기반 개발(Component-Based Development : CBD) 방법에 이르게 되었다[5]. 컴포넌트는 고유한 기능을 수행하는 독립적인 소프트웨어의 단위를 말하며, 인터페이스와 구현의 분리를 통해 캡슐화를 통하여 컴포넌트 제공자와 사용자 사이에서 독립성을 확보하여 소프트웨어의 재사용성을 높일 수 있게 하는 방법론이다. 컴포넌트 모델은 컴포넌트 설계와 구현 단계에서 표준 규약을 통하여 컴포넌트에 대한 일관성 있는 관리를 지원하며, 컴포넌트 패키징, 분산, 트랜잭션 관리, 통신, 보안 등의 서비스가 포함된다. 그러나 이러한 컴포넌트 모델의 분산 응용 프로그램을 운영하기 위하여 CORBA, DCOM, RMI 등이 개발되었지만 지속성있는 데이터를 표현하기 위한 표준화된 방법이 없었고, 트랜잭션, 보안, 멀티쓰레딩 등의 서비스를 위하여 개발자들이 직접 코드를 작성해야 하였다. 이러한 문제점들을 해결하기 위하여 현재 인정되고 있는 컴포넌

논문접수일 : 2008년 03월 07일 논문수정일 : 2008년 07월 22일 게재확정일 2008년 09월 01일
† 교신저자 mhlee@semyung.ac.kr

트 모델의 표준은 MS사의 COM+, OMG의 CCM(CORBA Component Model), SUN사의 EJB(Enterprise JavaBeans) 등이 있지만, 이 중에서 대용량 분산 객체의 가장 성공 모델로 알려진 것이 EJB이다. EJB는 자바 프로그램처럼 단독으로 실행되는 것이 아니라 EJB 컨테이너라는 소프트웨어에 설치되어야 실행될 수 있으며, EJB 컨테이너는 EJB 서버에 포함되어 있다.

현재까지 플랫폼의 변화에 따른 개발 생산성에 대한 비교 연구는 2가지의 애플리케이션에 대하여 다른 J2EE 플랫폼에서의 개발 생산성을 비교한 연구였으며, <표 1>과 같은 측정 항목을 통하여 비교한 연구 수준이다[10].

<표 1> J2EE 플랫폼별 개발 생산성 비교

애플리케이션 명	측정 항목	J2EE 1.4 플랫폼	Java EE 5 플랫폼	향상 수준
Adventure Builder	클래스 수	67	43	36% 클래스 감소
	LOC	3,284	2,777	15% LOC 감소
Roster App	클래스 수	17	7	59% 클래스 감소
	LOC	987	716	27% LOC 감소
	XML 파일 수	9	2	78% XML 파일수 감소
	XML LOC	792	26	97% XML LOC 감소

그러나 EJB 2.0과 EJB 3.0에 대하여 같은 플랫폼 상에서 소프트웨어 개발 생산성 비교에 대한 연구가 미비하였으며, 정량적 분석도 일부분의 LOC(Line of Code) 분석만 시도함에 따라 새로운 사양이 발표됨에도 구체적인 평가 지표와 지침이 부족하여 소프트웨어 생산성의 평가와 프로젝트의 새로운 시도에 제한이 있었다[6].

따라서 본 연구에서는 대용량 분산 객체의 가장 성공 모델인 EJB에 대하여 같은 플랫폼 상에서 EJB 2.0과 EJB 3.0 사양에 대하여 정량적인 평가 지표를 제시하여, 새로운 EJB 사양에 대한 정량적인 분석을 통하여 객관적인 소프트웨어 개발 생산성 연구에 대한 지침을 제공하고자 한다.

2. EJB의 기본 개념

2.1 EJB의 정의 및 특징

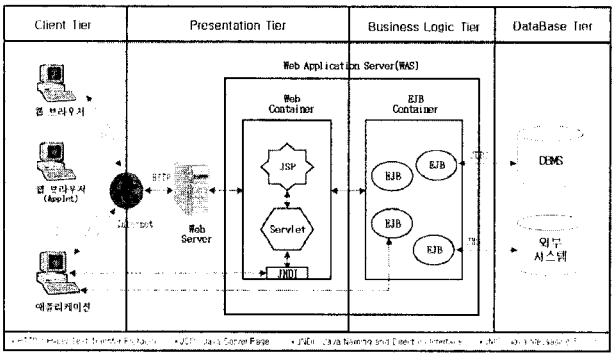
EJB는 N-tier의 분산형 객체지향 자바 애플리케이션을 개발하고 보급하기 위한 컴포넌트 구조로서 1998년에 SUN사에서 EJB 1.0 사양을 만들면서 J2EE 플랫폼의 핵심 컴포넌트로 자리 잡았다. 2000년에 EJB 1.1 사양의 발표를 통하여 1.0 스펙의 한계들을 극복함과 동시에 CORBA와의 호환성이 강조되었다. 2001년에 EJB 2.0 사양이 출시되면서 EJB는 인터넷 환경의 확산 및 자바 언어의 대중화에 따라 컴포넌트 개발을 위한 표준 플랫폼으로 자리잡아가고 있다. 이 버전은 본격적으로 대형 비즈니스 애플리케이션을 구축하는데 손색이 없을 정도의 기능을 제공하였다. 2006년에는 EJB 3.0 사양이 발표되면서 어떤 프레임워크에도 종속되지 않으며, 어떤 컨테이너에도 종속되지 않는 일반적인 자바 객체인 POJO(Plain Old Java Object) 기반의 코딩, 메타 데이터(Annotation), 엔티티빈의 변화(Persistence Model)를 통하여 시대의 요구사항에 맞춰 좀 더 유연하고, 테스트 및 개발하기 쉬운 새로운 EJB로 탄생하게 되었다[1, 7-9].

따라서 EJB는 확장성 있는 애플리케이션 서버 컴포넌트들을 지원하는 여러 서비스들을 제공함으로써 비즈니스 애플리케이션들을 컴포넌트 단위로 쉽게 작성할 수 있도록 되었다. EJB는 트랜잭션 처리 모니터(TP Monitor), 웹 서버, 데이터베이스 서버, 애플리케이션 서버 등과 같은 트랜잭션 처리 시스템에서 운영될 수 있다. EJB 컴포넌트 모델은 서버 컴포넌트들을 지원하기 위해 자바빈 컴포넌트 모델을 확장하였다. 이 서버 컴포넌트는 애플리케이션 서버에서 실행되는 애플리케이션 컴포넌트를 의미한다. 서버 컴포넌트들은 애플리케이션을 개발하기 위해 다른 컴포넌트들과 결합될 수 있다[4].

2.2 개발 환경

단독 애플리케이션 개발환경인 J2SE에서는 대용량 비즈니스 애플리케이션을 구현하기에는 부족한 점이 많아 트랜잭션, 보안, 메시징, 네이밍, 웹 등의 분산 애플리케이션을 작성할 수 있도록 N-tier 구조인 J2EE 플랫폼을 개발하게 되었다. 이 구조는 클라이언트-서버 구조에서 유연하게 대처하기 위하여 비즈니스 로직과 데이터베이스 처리 등의 개별적인 고유 애플리케이션을 서로 다른 시스템에 분산시킨 3-tier 구조에서 컴포넌트 구조를 좀 더 세분화시켜서 구현한 개념으로 이 구조부터 웹 기반 분산 객체 기반 애플리케이션이 운영되게 되었다[1-3].

따라서 본 연구에서 EJB2.0과 EJB 3.0 사양에서 개발 생산성을 비교 분석하기 위한 J2EE로 구현한 N-tier 개발 환경을 살펴보면 <그림 1>과 같다.



<그림 1> N-tier 분산객체의 구성도

3. 개발 생산성 비교 방안

3.1 테스트 환경

EJB 2.0과 EJB 3.0 사양에서의 발전은 <표 2>와 같이 객체지향 개념을 가진 실체 클래스(Entity Class)를 테이블로 대응시키는 객체관계매핑(Object Relational Mapping : ORM)에서 가장 많은 변화가 있다.

<표 2> ORM 부분에서의 차이점

EJB 2.0	EJB 3.0
◦ 엔티티 빈에서는 무거워서 다수의 데이터를 검색하는 경우 사용할 수 없는 성능을 보임. ◦ 반드시 하나의 테이블에서만 매핑되어야 함. ◦ 복잡한 생명주기로 인한 구현 부분의 어려움 발생. ◦ 쿼리문을 사용하는데 많은 제약이 있음.	◦ POJO 기반으로 가벼운 엔티티 모델을 추구함. ◦ 네이티브 쿼리문을 사용할 수 있게 하여 유연성을 확대함. ◦ 매핑하는데 소요되는 시간을 최소화함. ◦ 뛰어난 성능을 보장함. ◦ 매우 단순한 생명주기를 제공함. ◦ 높은 수준의 객체 지향 모델을 지원함.

이와 같이 본 연구에서는 EJB 2.0과 EJB 3.0 사양을 기반으로 하는 소프트웨어 개발 생산성을 비교 분석하기 위한 방안으로 <표 3>과 같은 동일한 개발환경과 데이터베이스 스키마를 이용하여 EJB 2.0과 EJB 3.0 환경에서의 파일럿 프로그램을 개발한 후, 이 프로그램 통하여 각 항목별 평가 지표를 이용하여 사양별 정량적으로 개발 생산성을 비교하도록 한다.

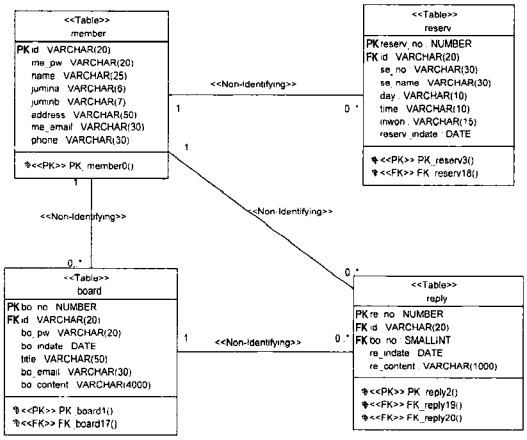
따라서 본 연구에서 사용한 정량적 소프트웨어 개발 생산성 평가 지표로 사양별 파일 개수의 비교, 세션빈과 엔티티빈의 LOC(Line Of Code) 비교, XML의 비교, 세션 타입 선언의 비교, 쿼리문의 비교, PK 선언의 비교, 콜백 메서드의 비교 등을 선정하여 분석한다.

<표 3> EJB 2.0과 EJB 3.0의 개발 환경

항 목	EJB 2.0	EJB 3.0
OS	Windows XP Professional	Windows XP Professional
Platform	J2SE 1.4, J2EE 1.3	J2SE 1.5, J2EE 1.4
WAS	JBOSS-4.0.5GA	JBOSS-4.0.5GA
DB	Oracle 10g	Oracle 10g
IDE	Eclipse 3.1	Eclipse 3.1
CASE	Rational Rose 2002	Rational Rose 2002

3.2 데이터베이스 스키마

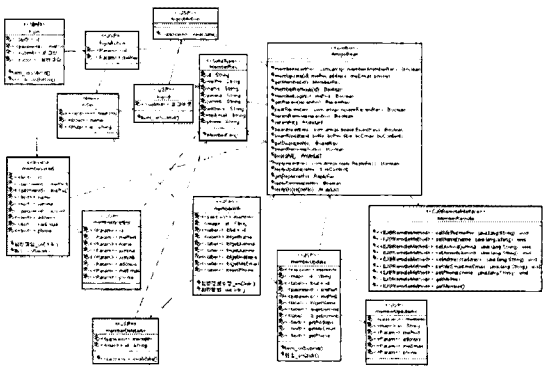
소프트웨어 생산성 비교를 위하여 개발될 파일럿 시스템의 데이터베이스 스키마는 EJB 2.0과 EJB 3.0 사양에서 <그림 2>와 같이 동일한 데이터베이스 스키마를 이용하여 비교 분석한다.



<그림 2> 데이터베이스 스키마 구조

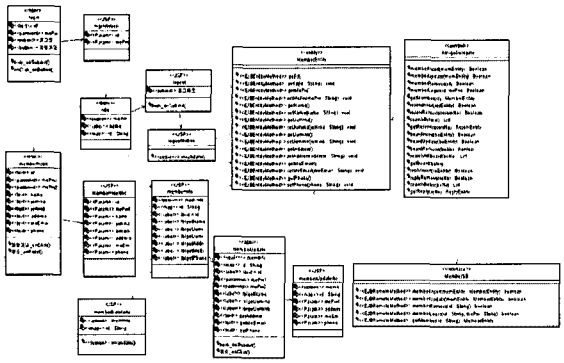
3.3 파일럿 시스템의 구현

이상과 같은 개발 환경과 데이터베이스 스키마를 기



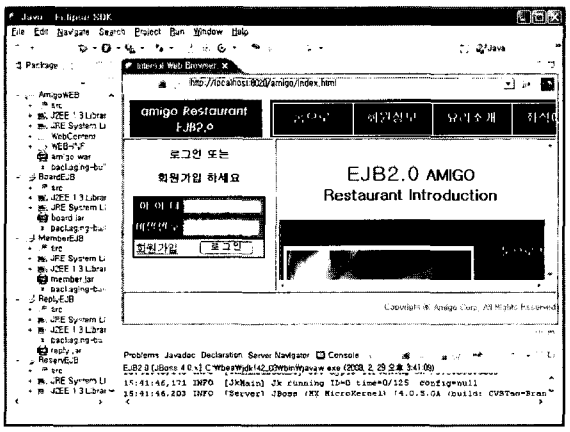
<그림 3> EJB 2.0의 클래스 다이어그램

반으로 EJB 2.0과 EJB 3.0의 개발 플랫폼에서 소프트웨어 개발 생산성을 정량적으로 분석하기 위한 구현된 파일럿 시스템의 회원관리 사례의 클래스 다이어그램 설계는 다음 <그림 3>과 <그림 4>이다.

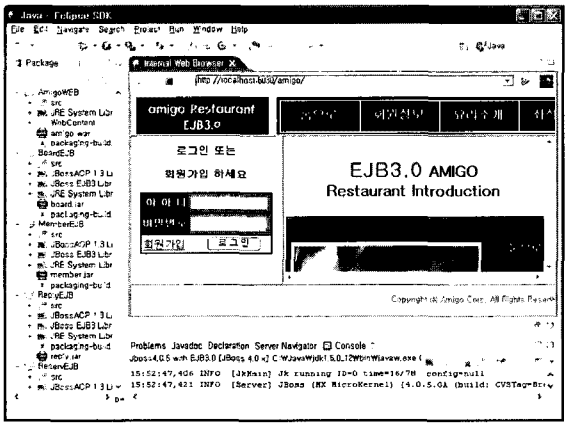


<그림 4> EJB 3.0의 클래스 다이어그램

따라서 위와 같은 분석 및 설계를 통하여 동일한 플랫폼 개발 환경에서 EJB 2.0과 EJB 3.0으로 구현된 파일럿 시스템은 다음 <그림 5>와 <그림 6>이다.



<그림 5> EJB 2.0의 파일럿 시스템



<그림 6> EJB 3.0의 파일럿 시스템

4. 평가 지표에 따른 비교 분석

4.1 파일 개수의 비교

정량적인 소프트웨어 평가지표 중에서 가장 기본적인 사양별 파일 개수를 비교하기 위하여 <그림 2>와 같은 데이터베이스 스키마 구조를 기반으로 EJB 2.0과 EJB 3.0의 파일럿 시스템을 구현하여 인터페이스 파일과 클래스 파일의 개수를 비교해 보면, EJB 2.0에서 세션빈의 개수는 12개가 필요하고, EJB 3.0에서는 8개만 있으면 구현이 가능하다. 또한 엔티티빈의 경우에는 EJB 2.0은 12개가 필요하지만 EJB 3.0은 4개만으로 구현이 가능하다. 따라서 EJB 3.0에서는 EJB 2.0보다 엔티티빈의 소프트웨어 생산성의 극대화를 볼 수 있다. 이상과 같은 결과를 요약하면 <표 4>와 같다.

<표 4> 인터페이스와 클래스 파일의 비교

항목	EJB 2.0		EJB 3.0	
	Session	Entity	Session	Entity
MEMBER	MemberSB MemberSBBean MenberSBHome	MemberBean MemberRemote MemberRemoteHome	MemberSB MemberSBBean	MemberEntity
BOARD	BoardSB BoardSBBean BoardSBHome	BoardBean BoardRemote BoardRemoteHome	BoardSB BoardSBBean	BoardEntity
RESERV	ReservSB ReservSBBean ReservSBHome	ReservBean ReservRemote ReservRemoteHome	ReservSB ReservSBBean	ReservEntity
REPLY	ReplySB ReplySBBean ReplySBHome	ReplyBean ReplyRemote ReplyRemoteHome	ReplySB ReplySBBean	ReplyEntity
계	12	12	8	4

4.2 LOC의 비교

4.2.1 세션빈의 LOC 비교

일반적으로 소프트웨어 개발 생산성을 비교할 때 LOC 비교 방법을 자주 사용한다. 따라서 본 연구에서도 동일한 플랫폼에서 EJB 2.0과 EJB 3.0 사양에서의 세션빈에 대한 LOC를 비교해 보면, <표 5>와 같이 사양별 LOC에 큰 차이는 없지만 EJB 3.0에서는 빈 클래스를 생성하고, 찾고, 삭제할 수 있는 메서드들을 선언하는 홈 인터페이스가 필요 없기 때문에 명시적인 복잡도가 줄어든 효과가 있다.

4.2.2 엔티티빈의 LOC 비교

EJB 2.0과 EJB 3.0 사양의 가장 많이 개선된 부분이 바로 엔티티빈 부분이다. EJB 3.0에서는 호출할 수 있

<표 5> 세션빈의 LOC 비교

항 목	파일명		코드량		비고 (LOC)
	EJB2.0	EJB3.0	EJB2.0	EJB3.0	
MEMBER	MemberSB	MemberSB	3	10	EJB2.0 : 51 EJB3.0 : 63
	MemberSBBean	MemberSBBean	41	53	
	MemberSBHome	-	7	-	
BOARD	BoardSB	BoardSB	3	11	EJB2.0 : 50 EJB3.0 : 60
	BoardSBBean	BoardSBBean	40	49	
	BoardSBHome	-	7	-	
RESERV	ReservSB	ReservSB	7	10	EJB2.0 : 73 EJB3.0 : 46
	ReservSBBean	ReservSBBean	59	36	
	ReservSBHome	-	7	-	
REPLY	ReplySB	ReplySB	3	10	EJB2.0 : 53 EJB3.0 : 46
	ReplySBBean	ReplySBBean	43	36	
	ReplySBHome	-	7	-	
Total(단위 : Line)			227	215	

는 메서드를 선언하는 리모트 인터페이스와 홈 인터페이스 부분이 필요 없기 때문에 LOC만 비교해 보면, <표 6>과 같이 EJB 3.0이 EJB 2.0에 비하여 67% LOC가 적은 것으로 나타났다. 또한 리모트 인터페이스와 홈 인터페이스의 구현이 필요하지 않아 소프트웨어를 개발하기 위한 복잡도 감소와 디버깅의 효율성이 가능하게 된 정성적인 효과도 있다.

<표 6> 엔티티빈의 LOC 비교

항 목	파일명		코드량		비고 (LOC)
	EJB2.0	EJB3.0	EJB2.0	EJB3.0	
MEMBER	MemberBean	MemberEntity	196	75	EJB2.0 : 221 EJB3.0 : 75
	MemberRemote		14		
	MemberRemoteHome		11		
BOARD	BoardBean	BoardEntity	190	73	EJB2.0 : 210 EJB3.0 : 73
	BoardRemote		11		
	BoardRemoteHome		9		
RESERV	ReservBean	ReservEntity	217	81	EJB2.0 : 241 EJB3.0 : 81
	ReservRemote		12		
	ReservRemoteHome		12		
REPLY	ReplyBean	ReplyEntity	186	57	EJB2.0 : 205 EJB3.0 : 57
	ReplyRemote		8		
	ReplyRemoteHome		11		
Total(단위 : Line)			877	286	591

4.3 XML의 비교

EAR 파일은 내부적으로 엔터프라이즈 빈을 위한 JAR 파일, 웹 응용프로그램을 위한 WAR 파일, 응용프로그램 클라이언트를 위한 JAR 파일 등을 포함하고 있다. 각 파일들은 내부적으로 자바 클래스 파일과 XML 문서 파일을 가지고 있다. 이 XML 파일을 배포서술자(Deployment Descriptor)라고 하며 엔터프라이즈 빈들에 대한 선언적인 정보를 가지고 있다. 이 배포서술자는 프로그램 코드가 아닌 트랜잭션 처리나 보안 처리, 네이밍 서비스와 같은 선언적인 정보를 이용하여 엔터프라이즈 빈의 행동을 정의할 수 있다.

본 연구에서 EJB 2.0과 EJB 3.0 환경의 파일럿 시스템 배포서술자의 비교표는 <표 7>과 같다.

<표 7> EJB 2.0과 EJB 3.0의 XML 비교표

분 류	배포서술자	EJB 2.0		EJB 3.0	
EJB	ejb-jar.xml	Board	42	X	
		Member	42		
		Reply	42		
		Reserv	42		
	persistance.xml	X		Board	12
				Member	12
				Reply	12
				Reserv	12
WEB	web.xml	13		13	
	jboss-web.xml	10		X	

위의 <표 7>과 같이 엔터프라이즈 빈의 행동을 정의하는 배포서술자는 EJB 3.0이 EJB 2.0보다 모든 빈들이 매우 단순해 졌으며, WAS 벤더들과 무관한 표준 사양으로 작성되어 소프트웨어의 독립성과 생산성을 확보하게 되었다.

4.4 세션타입 선언의 비교

EJB 2.0과 EJB 3.0의 다양한 세션 타입 선언에 대한 소프트웨어 생산성 비교가 가능하지만 본 연구에서는 무상태 세션빈(Stateless Session Bean)에 대하여 비교하도록 한다. EJB 2.0 사양에서는 무상태 세션빈은 ejb-jar.xml 배포서술자의 <session-type> 태그에 기술하지만, 이전의 EJB 개발의 어려움을 극복하기 위하여EJB 3.0 사양에서는 모든 빈을 POJO 기반으로 구현하도록 하고 있다.

따라서 EJB 2.0 이전에서 빈을 구현할 때마다 작성하였던 배포서술자와 상속 및 구현을 <표 8>과 같이 메타데이터(Annotation)인 @Stateless를 통해서 하도록 하였기 때문에 EJB 2.0에 비하여 매우 간소화되었다.

<표 8> 세션타입 선언의 비교표

EJB2.0 (ejb-jar.xml)	<session> <display-name>MemberSB</display-name> <ejb-name>MemberSB</ejb-name> <home>com.amigo.member.session.MemberSBHome</home> <remote>com.amigo.member.session.MemberSB</remote> <ejb-class>com.amigo.member.session.MemberSBBean</ejb-class> <session-type>Stateless</session-type> <transaction-type>Container</transaction-type> </session>

EJB3.0 (Member SBBean.java)	package com.amigo.member ;
	import javax.ejb.Stateless ;
	import javax.persistence.EntityManager ;
	import javax.persistence.PersistenceContext ;
	
	@Stateless
	public class MemberSBBean implements MemberSB {
	@PersistenceContext(unitName = "MemberEntityManager")
	private EntityManager em ;
	public MemberSBBean() {
	}

4.5 쿼리문의 비교

EJB 3.0의 데이터베이스 처리의 핵심인 검색 기능을 제공하는 쿼리 인터페이스와 EJB 엔티티빈에서 검색시 사용하는 모든 쿼리문은 com.j2eearchitect.ejb3.entity. Facade 무상태 세션빈에 정의되어 있으며, 이 세션빈은 엔티티빈을 처리하기 위한 퍼사드 역할을 수행한다.

이전의 EJB 2.0에서는 EJB 엔티티빈을 위한 쿼리 언어인 EJB QL을 사용하거나 BMP 엔티티빈에서 네이티브 SQL을 사용할 수 있었지만 많은 제약과 엔티티빈의 협업에 사용할 수 없는 단점이 있었다. 그러나 EJB 3.0 에서는 이러한 문제를 해결하기 위하여 새로운 네이티브 SQL을 지원한다. <표 9>는 EJB 3.0이 EJB 2.0과 비교하여 위의 문제를 해결하면서 매우 높은 소프트웨어 생산성의 쿼리문 구조를 지원하는 것을 알 수 있다.

<표 9> 쿼리문 비교

EJB2.0 (ReservSB Bean.java)	public Collection ejbFindById(String id) throws FinderException{
	String query = "select reserv_no from reserv where id=? order by reserv_no desc" ;
	Collection list = new ArrayList() ;
	try{
	Connection conn = ds.getConnection() ;
	PreparedStatement pstmt = conn.prepareStatement(query) ;
	pstmt.setString(1, id) ;
	ResultSet rs = pstmt.executeQuery() ;
	while(rs.next())
	list.add(new Integer(rs.getInt(1))) ;
	}catch(Exception e){
	e.printStackTrace() ;
	}
	return list ;
	}

EJB3.0 (Reserv Bean.java)	public List searchById(String id) {
	List list = em.createNativeQuery
	("select * from reserv where id = ? order by reserv_no desc", ReservEntity.class)
	.setParameter(1, id)
	.getResultList() ;
	return list ;
	}

4.6 PK 선언의 비교

EJB 2.0과 EJB 3.0의 PK(Primary Key) 선언 부분에 대한 소프트웨어 생산성을 비교해 보면, <표 10>과 같이 메타데이터를 통해서 EJB 2.0에 비하여 LOC가 매우 간소화되고 직관력이 높아진 것을 알 수 있다.

<표 10> PK 선언의 비교표

EJB2.0 (Member Bean.java)	public String ejbFindByPrimaryKey(String id) throws FinderException {
	String query = "select id from member where id = ?" ;
	try {
	Connection conn = ds.getConnection() ;
	PreparedStatement pstmt = conn.prepareStatement(query) ;
	pstmt.setString(1, id) ;
	ResultSet rs = pstmt.executeQuery() ;
	if (rs.next()) {
	return id ;
	}
	생 략
	. . .

EJB3.0 (Member Entity.java)	@Id
	@Column(name = "ID")
	public String getId() {
	return id ;
	}
	public void setId(String id) {
	this.id = id ;
	}

4.7 콜백 메서드의 비교

EJB에서 콜백 메서드는 WAS 서버가 각 빈들의 생명주기에 따른 객체 상태정보에 따라 호출하는 메서드를 말한다. 따라서 빈 클래스에 따라 반드시 구현해주어야 하는 콜백 메서드가 EJB 2.0에 비하여 EJB 3.0은 <표 11>과 같이 아주 단순해짐에 따라 소프트웨어 생산성 향상과 복잡성이 감소됨을 알 수 있다.

5. 결 론

현재까지 대용량 분산 객체 컴퓨팅 환경의 가장 성공 모델로 알려진 EJB에 대하여 표준 사양들이 지속적으로

<표 11> 콜백 메서드 비교

사 양	콜백 메서드	설 명
EJB 2.0	ejbCreate() ejbPostCreate()	빈 인스턴스를 생성하고 초기화하는 메서드
	ejbFind()	빈 인스턴스에 대해 조건에 맞는 데이터를 검색하는 메서드
	ejbRemove()	빈 인스턴스를 메모리에서 제거하는 메서드
	ejbLoad() ejbStore()	빈 인스턴스에 있는 컨테이너 관리 필드와 데이터베이스의 데이터를 일관성 있게 만드는 메서드
	ejbActivate() ejbPassivate()	빈 인스턴스를 풀에 둘 것인지 아니면 메모리 상에 띄울 것인지를 알려주는 메서드
	setEntityContext()	상태 정보를 저장하는 메서드
EJB 3.0	@PostConstruct	빈 인스턴스가 생성된 직후에 컨테이너가 호출하는 콜백 메서드
	@PreDestroy	빈 인스턴스를 삭제한 이후에 컨테이너가 호출하는 콜백 메서드
	@PostActivate	Passivate 된 빈 인스턴스가 activate 된 후에 호출하는 콜백 메서드
	@PrePassivate	빈 인스턴스가 passtivate 되어 2차 저장소에 저장되기 전에 호출하는 콜백 메서드

발표되고 있지만, 실무 프로젝트에서는 EJB의 기술 스킬 습득의 시간이 길고 표준 사양의 복잡도가 높음에 따라 쉽게 새로운 사양들을 현업에 적용하지 못하는 실정이다. 또한 EJB의 소프트웨어 개발 생산성 비교에 대한 연구도 사양의 일부분 LOC 분석만 시도하고 있으며, EJB의 새로운 사양이 발표됨에도 현재까지 구체적인 평가 지표 개발과 평가 지침이 부족하여 소프트웨어 생산성의 평가와 프로젝트의 새로운 시도에 제한이 있었다.

따라서 본 연구에서는 같은 플랫폼 상에서 EJB 2.0과 EJB 3.0 사양을 기반으로 파일럿 프로젝트를 개발하여 평가 지표를 제시하였으며, 새로운 EJB 사양에 대한 개발된 평가 지표에 따라 정량적인 분석을 통하여 객관적인 소프트웨어 개발 생산성 연구에 대한 지침을 제시하였다. 향후 Spring 프레임워크 상에서 EJB 3.0과의 소프트웨어 개발 생산성 비교와 J2EE N-tier 구조별 오픈 소스 플랫폼과의 평가 지표별 개발 생산성 비교에 대한 연구가 지속되어야 할 것이다.

참고문헌

[1] 김병곤; Enterprise Java Beans 3.0, 가메출판사, 26-340, 2006.

[2] 이명호; "X-인터넷 환경에서 생산성 있는 소프트웨어 개발 방법의 설계", 한국산업경영시스템학회지, 28(2) : 53-59, 2005.

[3] 이명호; "설비정보화를 위한 MVC 디자인 패턴을 활용한 비즈니스 티어의 설계", 대한설비관리학회지, 11(2) : 99-107, 2006.

[4] 전혜영, 김성진; 웹로직으로 배우는 EJB, 도서출판 대림, 12-301, 2006.

[5] 채홍석; 객체지향 CBD 개발 Bible, 한빛미디어, 35-76, 2006.

[6] Debu Panda, Reza Rahman, and Derek Lane; EJB 3 in Action, Manning Publications Co., 3-176, 2007.

[7] Richard Monson-Haefel, Bill Burke; "Enterprise Java-Beans 3.0," O'Reilly, 1-150, 2006.

[8] Sun Microsystems; "Enterprise JavaBeans Specification, Version 2.0," Final Release, 2001.

[9] Sun Microsystems; "JSR220 : Enterprise JavaBeans, Version 3.0," Final Release, 2006.

[10] John Steams, Roberto Chinnici, and Sahoo; "An Introduction to the Java EE 5 Platform," http://java.sun.com/developer/technicalArticles/J2EE/intro_ee5/index.html, 2006.