DbC 접근법을 이용한 EJB 기반 애플리케이션의 명세 기법*

(Specification Technique of EJB-Based Application using Design by Contracts Approach)

노 혜 민 † 유 철 중 ††
(Hye-Min Noh), (Cheol-Jung Yoo)

요 약 분산 웹 애플리케이션에 대한 관심도가 증가함에 따라서 복잡한 분산 프레임워크와 관련된 코드의 작성 없이 비즈니스 코드 작성에 주력할 수 있게 해 주는 서버측 Java 컴포넌트 아키텍처인 EJB에 대한 관심도 또한 증가하고 있는 추세이다. 그러나 이러한 관심도의 증가에도 불구하고 지금까지는 이러한 시스템의 신뢰성 측면에 대한 노력은 미비한 것이 현실이다. 따라서 본 논문에서는 객체지향 시스템 개발에서 소프트웨어의 신뢰성을 높여줄 수 있는 접근법으로 증명되고 있는 DbC 접근법을 EJB 기반 애플리케이션의 정형 명세 작성에 적용하기 위한 명세 기법을 제안한다. 이러한 명세 기법을 통해 개발자들은 EJB 기반 애플리케이션 개발에 있어서 신뢰성 측면의 이점을 얻을 수 있다.

키워드 : DbC(Design by Contract), 정형명세, EJB

Abstract Due to increased concern about the distributed web application, the interest in EJB - server-side Java component architecture that enables to make out Business Logic without writing codes related to complicated distributed framework - is also increasing. Despite of these increased interest, However, efforts for reliability of these systems have been insufficient. Thus, in this paper, we propose specification technique for applying DbC approach, which can elevate the reliability of software in the Object-Oriented system development, in writing formal specification of EJB-based application. Through this specification technique, developers can gain reliability in the EJB-based application development.

Key words : DbC(Design by Contract), Formal Specification, EJB

1. 서 론

컴포넌트기반 개발(Component-Based Development, 이하 CBD)은 요구사항 분석, 설계, 구현, 테스트, 배치 및 기타 기술적인 지원 등을 포함하는 개발 라이프사이클의 모든 측면 및 단계를 컴포넌트에 기반을 두고 있는 소프트웨어 개발 접근법이다[1]. 이러한 CBD 방법론은 현재 꾸준히 소프트웨어 개발 분야에서 성장해 가고 있으며, CBD 사용자 및 예찬론자들에 의해 여러 가지 방법론들이 쏟아져 나오고 있다[2]. 그러나 CBD 방법론에 대한 많은 연구에도 불구하고 신뢰성 있는 설계 문제에 대해서는 많은 노력이 기울여지고 있지 않고 있다[3].

신뢰성 있는 소프트웨어의 구축은 소프트웨어공학의 궁극적인 목적이다[4]. 신뢰성은 정확성(correctness)과 견고성(robustness) 있는, 간단히 말해서 버그가 없는 소프트웨어를 나타낸다[2,5]. 컨트랙트에 의한 설계(Design by Contract, 이하 DbC) 접근법은 소프트웨어의 정확성을 표현하고 확인하기 위한 필수적인 도구를 제공함으로써 '신뢰성 있는 소프트웨어의 구축'이라는 대전제를 직접적으로 다루고 있는 기법이며[4], 기존의 객체지향 개발 방법론에 적용되어 많은 이점을 가져온 것으로 증명되었다[2,5,6]. 최근 UML에서도 OCL

· 본 연구는 한국과학재단 목적기초연구(R05-2001-000-01033-0) 지원으로 수행되었음
† 학생회원 : 전북대학교 대학원 컴퓨터통계정보학과
hmino@cs.chonbuk.ac.kr
†† 종신회원 : 전북대학교 자연과학대학 컴퓨터과학과 교수
cjyoo@moak.chonbuk.ac.kr
논문접수 : 2002년 2월 14일
심사완료 : 2002년 9월 25일

(Object Constraint Language)을 추가함으로써 이러한 DbC 접근법을 지원하고 있다[7]. 이러한 배경을 고려해 볼 때 CBD 방법론을 통한 EJB 기반 애플리케이션의 구축에 신뢰성 측면의 이점을 얻기 위해서는 DbC 접근법의 적용이 하나의 해결책이 될 수 있다.

EJB 기반 애플리케이션 구축에 DbC 접근법을 적용할 필요성은 현재 선호되고 있는 시스템의 설계 스타일에서도 찾을 수 있다. Yu Lui는 현재 선호되는 시스템의 설계 스타일이 무엇인지에 대한 물음에서 DbC 적용의 필요성을 강조하였다[8]. 그는 시스템의 설계 스타일을 요구 스타일(demanding style)과 허용 스타일(tolerant style)로 구분하였다. 요구 스타일은 서비스 설계자가 서비스가 정확히 동작하는 조건만을 기술해 놓고 이를 사용하는 사용자가 이러한 전제조건을 따르게 하는 스타일을 말하며, 허용 스타일은 서비스 설계자가 가능한 모든 상황에서 서비스가 정확하게 동작한다는 것을 보증할 수 있도록 설계하는 스타일을 말한다. 현재는 허용 스타일보다는 요구 스타일이 선호되고 있다[8]. 그 이유는 최근의 다변화된 시스템 환경에서 어떠한 설계자도 시스템이 사용될 환경이나 미래의 사용에 대해서는 예측할 수 없기 때문에 시스템이 정확하게 동작하는 환경만을 명세해주는 요구 스타일이 더욱 효과적일뿐만 아니라 신뢰성 있는 소프트웨어를 구축하기 위한 설계 스타일임이 분명하다[8]. 이는 엔터프라이즈 빈즈라는 컴포넌트들의 조립을 통해 시스템을 구축하는 EJB 기반 애플리케이션에서는 더더욱 그렇다. 이러한 관점에서 사전조건, 사후조건, 불변조건으로 구성되는 DbC 접근법은 요구 스타일의 설계에 부합되는 기법이며, 이는 DbC 접근법을 EJB 기반 애플리케이션 구축에 적용해야 할 필연성을 뒷받침 해 준다.

본 논문의 목적은 객체지향 시스템에서 정확성과 견고성을 높여줄 수 있는 접근법으로 증명되고 있는 DbC 접근법을 EJB 기반 애플리케이션 구축에 적용하기 위한 명세 기법을 제안함으로써 정확성과 견고성이 높은 즉, 신뢰성 있는 소프트웨어를 구축하기 위함이며, OCL 명세의 기반이 되는 EJB 기반 애플리케이션의 그래픽 모델 작성 방법에서부터 OCL 명세를 작성하기 위한 명세 규칙의 제안까지를 연구 범위로 하고 있다. 그림 1은 이러한 범위의 연구를 수행하기 위해 진행한 작업 단계와 각각의 단계에서의 연구 내용을 간단히 보여주고 있다. 이러한 연구를 위해 현재 CBD 방법론의 한 가지로서 분산 웹 애플리케이션 구축 시 복잡한 분산 프레임워크와 관련된 코드의 작성 없이 비즈니스 코드 작성에 주력할 수 있도록 해 주는 서버측 Java 컴포넌트 아키텍처인 EJB[9]를 분석하여, 객체지향 방법론에서 신뢰성 있는 소프트웨어 구축 기법으로 적용되어온[10,11] DbC 접근법을 EJB 기반 애플리케이션 구축에 적용할 수 있도록 필요한 가정, 정의, 규칙들을 규명해 보고 이에 따라 실제 EJB 기반 애플리케이션을 명세해 본다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 본 연구의 근간이 되는 신뢰성 있는 소프트웨어 구축을 지원하는 DbC 접근법에 대하여 좀더 세부적으로 알아보고, DbC 접근법이 실제 적용된 사례를 기술한다. 3장에서는 관련연구를 토대로 DbC를 적용한 EJB 기반 애플리케이션의 명세 기법을 제안한다. 4장에서는 3장에서 제안한 명세 기법을 토대로 실제 EJB 기반 애플리케이션을 명세해 본다. 마지막으로, 5장에서는 결과와 향후 연구과제를 제시한다.

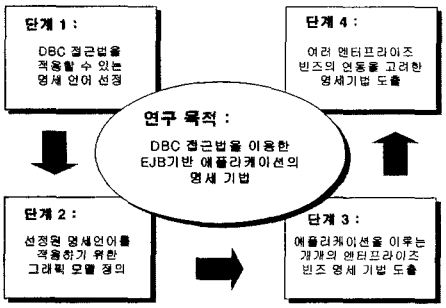


그림 1 연구 진행 단계

2. DbC 접근법

클래스와 인터페이스 그리고 그 클래스의 오퍼레이션을 명세하는 효과적인 방법은 사전조건과 사후조건 그리고 불변조건을 사용하는 것이다[12]. 이러한 사전조건, 사후조건, 불변조건을 이용하는 명세 방법을 DbC 접근법이라 한다.

이러한 DbC 접근법의 개념은 현재 체계적인 객체지향 소프트웨어 구축 접근법의 중심에 있으며, 이는 Eiffel 방법론[10]으로 구체화되어 있다[5]. DbC 접근법에서 컨트랙트의 정의는 일상에서의 계약 개념으로부터 유도해 낼 수 있다[12]. 일상에서의 계약은 두 당사자들간의 책임과 권리를 법적으로 명시하여 이를 서로 동의하여 성사된 것을 말한다. DbC 접근법에서의 컨트랙트도 이와 유사한 개념이다. 즉, 객체의 책임을 분명하고 모호하지 않게 명시해 놓은 것을 컨트랙트라고 하며, 서비스를 사용하는 모든 객체들을 클라이언트라고

하고, 서비스를 제공하는 객체를 공급자라고 할 때, 공급자는 클라이언트가 특정 조건을 만족시킬 경우에만 어떠한 서비스를 제공한다는 개념이다.

비록 컨트랙트의 개념이 법률적인 행위로부터 나왔다 하더라도 실제 소프트웨어 구축에 적용되었을 때에는 계약과 완전히 같은 개념은 아니다[12]. DbC 접근법에서의 컨트랙트는 클라이언트의 존재와는 독립적으로 객체에 의해 제공되며, 클라이언트는 그 컨트랙트에 있는 조건을 지킬 의무가 있는 것이다.

2.1 컨트랙트의 내용

컨트랙트는 객체에 의해 제공되는 서비스들에 대하여 기술한다. 다음은 각각의 서비스들에 대해 컨트랙트가 기술하는 내용이다[12].

- 서비스가 제공되기 위한 조건
- 서비스가 제공되기 위한 조건이 만족되었을 경우 서비스가 제공된 후 결과에 대한 명세

컨트랙트의 내용을 살펴보기 위해 다음 예를 살펴본다.

인천국제공항에서는 최소한 탑승시간 20분전까지 공항에 도착하고, 선적이 허용된 화물을 소지하고 있으며, 비행기 운임을 지불한 고객에 한해서 LA로의 비행을 허락한다.

컨트랙트는 모호하지 않도록 공급자와 클라이언트의 책임 및 권한에 대한 내용을 기술한다. 이러한 기술 내용을 소프트웨어 용어로 정형명세라고 하며[12] 표 1은 위의 예를 컨트랙트 형태로 표현한 것이다.

표 1 컨트랙트에서의 책임 및 권한

	책 임	권 리
클라이언트	• 인천국제공항에 탑승시간 20분전까지 도착해야함 • 소지한 화물은 선적 가능해야함 • 비행기 운임을 지불해야 함	• LA에 안전하게 도착함
공급자	• 클라이언트를 LA까지 안전하게 수송해야 함	• 탑승시간 5분전까지 인천국제공항에 도착한 클라이언트만 승선시킴 • 선적이 허용된 화물을 소지한 클라이언트만 승선시킴 • 비행기 운임을 지불한 클라이언트만 승선시킴

표 1에서 클라이언트의 책임은 공급자의 권리가 되며 이는 서비스가 제공되기 위해 필요한 조건이며, 클라이언트의 권리 즉, 공급자의 책임은 서비스가 제공되기 위한 조건이 만족되었을 경우 서비스가 제공된 후의 결과를 나타낸다.

분명하게 정의된 컨트랙트는 서비스의 클라이언트와 공급자 모두에게 다음과 같은 이점을 줄 수 있다[12].

- 공급자는 자신의 서비스가 사용될 수 있는 명확한 조건을 알 수 있다. 만약 클라이언트가 이와 같은 조건 즉, 클라이언트의 책임을 만족시키지 못한다면 공급자는 그 결과에 대한 책임이 없다. 이것은 공급자가 항상 만족되어야 하는 조건 하에서만 서비스를 할 수 있도록 해 준다.
- 클라이언트는 제공되는 서비스를 사용하기 위한 조건과 그 조건을 충족시켰을 경우 받을 수 있는 서비스 결과를 명확히 알 수 있다. 만약 클라이언트가 그 조건을 충족시킨다면 정확한 서비스의 실행을 보장받을 수 있다는 것을 의미한다.

만약 클라이언트가 명시된 조건을 충족시키지 못하거나 공급자가 명시된 서비스를 정확히 수행하지 못한다면 그 컨트랙트는 깨지게 된다. 따라서 컨트랙트를 통해 작성된 소프트웨어의 정형명세는 책임 및 권한을 명확히 해 줌으로써 신뢰성 있는 소프트웨어를 구축할 수 있게 해 준다.

2.2 사전조건, 사후조건, 불변조건

표 1에서 들고 있는 예를 하나의 객체가 제공하는 오퍼레이션으로 간주했을 때 공급자의 권리 즉, 클라이언트의 책임은 컨트랙트에서 사전조건으로 명세된다. 사전조건은 그 오퍼레이션이 실행되고 있는 순간 항상 참이어야 한다. 반대로 공급자의 책임 즉, 클라이언트의 권리는 사후조건으로 명세된다. 사후조건은 오퍼레이션이 그 실행을 마친 후 반드시 참이어야 하는 조건들을 말한다.

Bertrand Meyer는 사전조건과 사후조건 이외에 불변조건이라는 제약사항을 정의하였다[13]. 불변조건은 항상 클래스와 연결된다. 불변조건이란 클래스의 모든 인스턴스들이 항상 유지해야하는 조건을 말한다. 불변조건은 모든 시간에 항상 참이어야 하는 조건이다. 이는 사전조건 및 사후조건이 오퍼레이션 실행 전후 즉, 특정 시간에 참이어야 조건과 대조되는 것이다.

DbC 접근법은 이러한 사전조건, 사후조건, 불변조건으로 소프트웨어를 명세하는 접근법을 말하며 OCL과 같은 형식 언어를 이용한 명세를 정형 명세라고 한다.

2.3 DbC 접근법의 이점

DbC 접근법의 이점은 많은 문헌에서 찾아볼 수 있다. 그 중 ISE(Interactive Software Engineering)는 DbC 접근법의 이점을 다음과 같이 기술하였다[5].

- 객체지향 방법론뿐만 아니라 더 나아가 소프트웨어 구축에 대한 이해력을 높여준다.
- 버그 없는 객체지향 시스템 구축을 위한 체계적인 접근법이다.
- 디버깅, 테스팅, 더 나아가 품질 보증을 위한 효과적인 프레임워크이다.
- 소프트웨어 컴포넌트의 문서화를 위한 훌륭한 방법론이다.
- 비정상적인 상황을 다루기 위한 기법으로 예외 처리를 위한 안전하고 효과적인 방법론이다.

또한 Benoit, Traon, Jézéquel은 진단 가능성(diagnosability)이라는 소프트웨어 품질 척도를 통해 DbC 접근법의 우수성을 기술하였다[14]. 이들은 진단 가능성을 설명하기 위해 진단 범위(diagnosis scope)라는 용어를 정의하였다. 진단 범위는 소프트웨어에 결함이 발생하여 그 결함이 실패로 이어지거나 검출되었을 경우 테스트를 수행해야 되는 실행 스레드의 길이를 말한다. 따라서 진단 범위가 짧을수록 진단 가능성은 높아지게 되며 진단 가능성이 높을수록 고품질의 소프트웨어임을 나타낸다. 즉, DbC 접근법은 소프트웨어 컴포넌트의 제약사항을 명확히 명세함으로써 테스팅에 소요되는 노력 또한 줄일 수 있다.

요컨대 DbC 접근법은 초창기 객체지향 방법론과 결합되어 소프트웨어 구축에 있어 많은 이점을 가져온다는 것이 증명되어 왔다[3,5,6,14]. 이러한 이점에 힘입어 최근 DbC 접근법은 소프트웨어 구축 전반에 걸친 여러 방법론에 적용이 시도되고 있다.

3. DbC 접근법을 이용한 EJB 기반 애플리케이션의 명세 기법

본 장은 다음과 같이 구성되어 있다. 먼저 본 장에서 제안하고 있는 명세 규칙 각각에 대한 실제 명세 사례를 각 규칙과 함께 보이기 위해 명세 대상으로 이용할 애플리케이션 예제에 대하여 간략히 소개한다. 그리고 난 후 명세 기법의 근간이 되는 기초 개념과 더불어 몇 가지 그룹으로 분류하여 명세 규칙을 제안하고 각 명세 규칙에 대한 실제 적용사례를 서두에서 언급한 예제를 이용하여 제시한다. 마지막 절에서는 명세 규칙을 적용하여 명세하기 위한 명세 단계를 제시한다.

3.1 명세 예제 : 도서 대출 처리 애플리케이션

명세 대상은 도서관의 도서 대출을 처리해 주는 도서 대출 처리 애플리케이션으로 도서 대출 계정 정보를 가지고 있는 LibraryAccout 엔티티 빈과 도서관 이용자에 의해 발생할 수 있는 비즈니스 로직을 처리하는 LendingClerk 세션 빈, 도서 정보를 가지고 있는 Book 엔티티 빈과 도서관 정보를 가지고 있는 Library 엔티티 빈으로 구성되어 있다. 각 빈에 대한 개략적인 정보는 표 2에 나타나 있다.

3.2 기초 개념

DbC 방법론은 서론에서도 언급하였던 것처럼 객체지향 방법론에서 출발한 기법이다. 즉, 객체지향 방법론이 이슈화 될 때 그래픽 모델링 방법을 보완할 수 있도록 제안된 제약사항 명세 기법이다[4]. UML의 OCL도 이와 같은 배경에서 객체의 제약사항을 명세할 수 있도록 버전 1.1에서부터 추가되었다. 즉, DbC 접근법이나 OCL 언어는 모두 객체지향 방법론을 기반으로 하고 있다. 그 단적인 예가 DbC 접근법에서 사전조건, 사후조건, 불변조건의 기술 모두 객체지향 방법론에서 클래스와 클래스의 오퍼레이션을 기반으로 한다는 것이며, UML의 OCL 역시 UML의 클래스다이어그램을 기반으로 기술하는 명세언어이다.

따라서 현재 쉽고 빠른 애플리케이션 개발을 위해 많은 이슈가 되고있는 CBD 방법론을 기반으로한 EJB 기반 애플리케이션 구축[15]에 객체지향 방법론을 기반으로 하고 있는 DbC 접근법 및 OCL 언어를 완벽히 적용하기에는 무리가 있다. 그 이유는 객체지향 개발 방법론과 엔터프라이즈 빈즈라는 컴포넌트를 조립하여 애플리

표 2 도서 대출 처리 애플리케이션의 빈 구성

빈 이름	빈 타입	기능
LibraryAccount	엔티티 빈	계정 id, 계정 소유자 이름, 도서 대출 한도, 계정의 대출 도서 부수 등의 정보를 지니고 있으며, 이와 같은 도서관 대출 계정 관련 데이터의 처리 기능을 담당한다.
LendingClerk	세션 빈	특정 계정의 도서 대출, 도서 대출 연장, 도서 대출 예약 등의 비즈니스 로직 부분을 담당한다.
Book	엔티티 빈	고유의 도서 바코드 번호, 도서 제목, 예약 여부, 대출 가능 여부 등의 정보를 지니고 있으며, 이와 같은 도서와 관련된 데이터의 처리 기능을 담당한다.
Library	세션 빈	소장 도서 목록 등의 도서관과 관련된 데이터의 처리 기능을 담당한다.

케이션을 구축하는 일종의 컴포넌트기반 개발 방법론과는 차이점이 존재하기 때문이다[16]. 따라서 DbC 접근법을 EJB 기반 애플리케이션 구축에 적용하기 위한 노력이 필요하며, 이를 통해 신뢰성 있는 EJB 기반 애플리케이션을 구축할 수 있다.

따라서 본 장에서는 객체지향 방법론에 적용되어온 DbC 접근법을 지원하는 UML의 OCL 언어를 EJB 기반 애플리케이션의 명세에 적용 및 이용하기 위한 명세 규칙을 제안한다. 이를 위해 명세 규칙을 몇 가지 범주로 구분하여 각각의 규칙 설정에 필요한 가정 및 정의를 한 후 명세 규칙을 제안한다. 또한 앞 절에서 소개한 도서 대출 처리 애플리케이션을 통해 제안된 명세 규칙의 적용 사례를 소개한다.

다음은 본 장에서 제안하는 모든 명세 규칙의 기반이 되는 기초 가정들이다.

(가정 1) 명세의 기반이 되는 그래픽 모델은 EJB 컴포넌트 클래스 다이어그램이라 한다.

(가정 2) 사전조건, 사후조건, 불변조건의 명세 대상은 엔터프라이즈 빈즈의 빈 클래스로 한정한다.

3.3 EJB 컴포넌트 클래스 다이어그램 작성 규칙

본 절에서는 OCL 언어의 명세 기반이 되는 그래픽 모델인 EJB 컴포넌트 클래스 다이어그램의 작성 규칙을 제안한다. OCL은 본래 UML의 클래스 다이어그램을 기반으로 명세를 작성하는 언어이다. 따라서 OCL을 이용하기 위해서는 UML의 클래스 다이어그램과 같은 형태의 그래픽 모델이 있어야 하며, 본 논문에서는 ***(가정 1)***에서 이를 EJB 컴포넌트 클래스 다이어그램이라 가정하였다. EJB 기반 애플리케이션은 엔터프라이즈 빈즈라는 컴포넌트들로 구성된 애플리케이션이며, 또한 각각의 엔터프라이즈 빈즈는 EJB 명세서에 정의된 몇몇 클래스들로 구성된다. 따라서 그래픽 모델인 EJB 컴포넌트 클래스 다이어그램은 이러한 모든 정보를 지니고 있어야 한다. 그러나 애플리케이션을 구성하는 엔터프라이즈 빈즈들간의 정보와 각각의 엔터프라이즈 빈즈를 구성하는 클래스간의 정보 모두를 하나의 그래픽 모델에 표현한다면 추상화 수준이 너무 낮아져 그래픽 모델의 가독성을 해칠 수 있다. 따라서 본 논문에서는 EJB 컴포넌트 클래스 다이어그램을 EJB 컴포넌트 클래스 개요 다이어그램과 EJB 컴포넌트 클래스 상세 다이어그램 두 가지 형태로 작성할 것을 제안한다. EJB 컴포넌트 클래스 개요 다이어그램은 엔터프라이즈 빈즈들간의 연관성 및 다중성만을 개략적으로 모델링한 다이어그램이며, EJB 컴포넌트 클래스 상세 다이어그램은 개요 다이어그램의 일부를 각각의 엔터프라이즈 빈즈를 구성하는 세부 클래스들의 정보를 나타내는 엔터프라이즈 빈즈 클래스 모델로 표현한 다이어그램을 말한다.

3.3.1 EJB 컴포넌트 클래스 개요 다이어그램 작성 규칙

EJB 컴포넌트 클래스 개요 다이어그램은 일반적인 컴포넌트 다이어그램과 동일하게 작성할 것을 제안하며, 따라서 특별한 가정과 정의는 필요하지 않다. 작성 규칙은 다음과 같다.

[규칙 1] EJB 컴포넌트 클래스 개요 다이어그램은 엔터프라이즈 빈즈의 세부 사항보다는 엔터프라이즈 빈즈 간의 연관관계를 강조한 다이어그램으로 그림 2와 같은 형태로 작성한다.

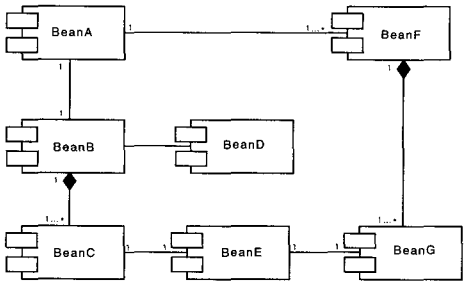


그림 2 EJB 컴포넌트 클래스 개요 다이어그램

앞서 규정한 EJB 컴포넌트 클래스 개요 다이어그램 작성 규칙(***[규칙 1]***)의 적용 사례를 살펴보기 위해 도서 대출 처리 애플리케이션을 고려해 보자. 그림 3은 ***[규칙 1]***에 따라 작성한 도서 대출 처리 애플리케이션의 EJB 컴포넌트 클래스 개요 다이어그램을 보여준다.

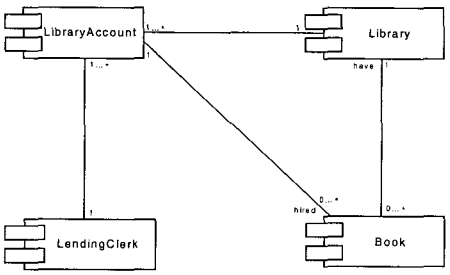


그림 3 도서 대출 처리 애플리케이션의 EJB컴포넌트 클래스 개요 다이어그램

3.3.2 EJB 컴포넌트 클래스 상세 다이어그램 작성 규칙

EJB 컴포넌트 클래스 상세 다이어그램은 개개의 엔터프라이즈 빈즈를 구성하는 클래스 정보들을 상세하게 표현한 엔터프라이즈 빈즈 클래스 모델을 통해 구성되며, 엔터프라이즈 빈즈 클래스 모델은 세션 빈 클래스 모델과 엔티티 빈 클래스 모델로 분류하였다. EJB 컴포넌트 클래스 상세 다이어그램을 작성하기 위해 다음과 같이 가정한다.

(가정 3) 세션 빈 클래스 모델은 연관성 및 다중성을 고려한 홈 인터페이스, 리모트 인터페이스, 빈 클래스로 구성되며, 모델 작성 시 리모트 인터페이스에 정의된 오퍼레이션은 빈 클래스의 오퍼레이션과 동일하므로 생략한다.

(가정 4) 엔티티 빈 클래스 모델은 연관성 및 다중성을 고려한 홈 인터페이스, 리모트 인터페이스, 빈 클래스, 프라이머리 키 클래스, 데이터베이스 레코드로 구성되며, 모델 작성 시 리모트 인터페이스에 정의된 오퍼레이션은 빈 클래스의 오퍼레이션과 동일하므로 생략한다.

(가정 5) 엔티티 빈 클래스 모델과 연관되는 데이터베이스의 테이블은 하나의 클래스로 간주하여 빈 클래스 모델 내에 삽입하고 데이터베이스의 레코드들은 테이블(클래스)의 인스턴스들(객체)로 간주한다.

이러한 가정들은 빈 클래스, 홈 인터페이스, 리모트 인터페이스, 프라이머리 키 클래스 등으로 구성되는 하나의 엔터프라이즈 빈즈를 연관관계를 갖는 객체들로 인식하고, 이러한 엔터프라이즈 빈즈들간의 연관관계를 다시 고려함으로써 객체지향 방법론에 적용되어온 OCL 언어를 엔터프라이즈 빈즈들로 구성되는 EJB 기반 애플리케이션을 명세하는데 적용할 수 있도록 그래픽 모델을 구성하기 위함이다.

<정의 1> 엔터프라이즈 빈즈 클래스 모델에서 빈 클래스는 홈 인터페이스 및 리모트 인터페이스와 연관성을 가지며 다중성은 1 : 1이다.

<정의 2> 엔티티 빈 클래스 모델에서 빈 클래스는 프라이머리 키 클래스와 연관성을 가지며 다중성은 1 : 1이다.

<정의 3> 엔티티 빈 클래스 모델에서 빈 클래스와 데이터베이스 레코드들간에는 연관성을 가지며 다중성은 1 : 1...* 이다.

<정의 1>은 가정들을 적용했을 경우 엔터프라이즈 빈즈 클래스 모델에서 빈 클래스와 홈 인터페이스, 리모트 인터페이스와의 연관성 및 다중성을 정의하고 있다. ***<정의 2>***와 ***<정의 3>***은 엔티티 빈 클래스 모델에서 빈 클래스와 프라이머리 키 클래스, 데이터베이스 레코드들간의 다중성을 정의하고 있다. ***<정의 1>*** ~ ***<정의 3>***은 DbC 접근법을 적용할 OCL 명세의 근거가 되는 정의로서, 명세에서 OCL 표현식의 작성 시 사전 정의된 OCL 타입의 사용 및 OCL 문법의 적용 근거가 된다.

앞서 기술한 가정과 정의를 적용한 EJB 컴포넌트 클래스 상세 다이어그램의 규칙은 다음과 같다.

[규칙 2] 세션 빈 클래스 모델은 ***(가정 3)***, ***<정의 1>***에 의해 그림 4와 같은 형태로 작성한다.

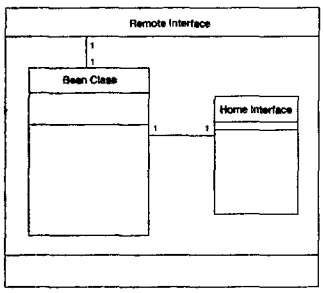


그림 4 세션 빈 클래스 모델

[규칙 3] 엔티티 빈 클래스 모델은 ***(가정 4)***, ***(가정 5)***, ***<정의 1>***, ***<정의 2>***, ***<정의 3>***에 의해 그림 5와 같은 형태로 작성한다.

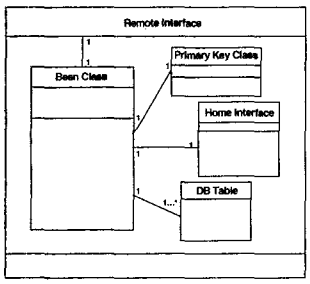


그림 5 엔티티 빈 클래스 모델

[규칙 4] EJB 컴포넌트 클래스 상세 다이어그램은 ***[규칙 2]***, ***[규칙 3]***에 의해 작성한 엔터프라이즈 빈즈 클래

스 모델을 이용하여 그림 6과 같은 형태로 작성한다.

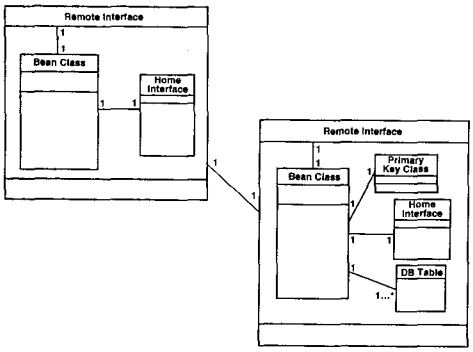


그림 6 EJB 컴포넌트 클래스 상세 다이어그램

앞서 규정한 EJB 컴포넌트 클래스 상세 다이어그램 작성 규칙(*[규칙 2], [규칙 3], [규칙 4]*)의 적용 사례를 살펴보기 위해 도서 대출 처리 애플리케이션을 고려해 보자. 그림 7은 *[규칙 2], [규칙 3], [규칙 4]*에 따라 작성한 도서 대출 처리 애플리케이션의 EJB 컴포넌트 클래스 상세 다이어그램을 보여준다.

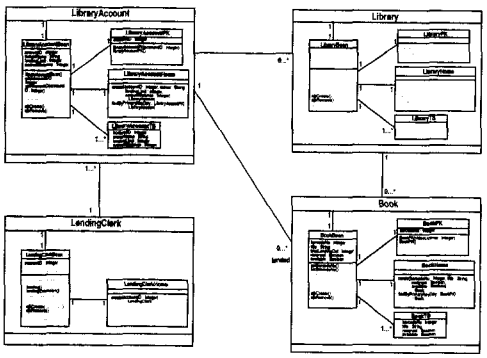


그림 7 도서 대출 처리 애플리케이션의 EJB 컴포넌트 클래스 상세 다이어그램

3.4 불변조건 작성 규칙

불변 조건은 EJB 컴포넌트 클래스 다이어그램을 기반으로 일반적인 OCL 작성 규칙에 따라 작성한다. 따라서 가정은 필요하지 않다. 다음은 EJB 기반 애플리케이션에서 불변조건이 가지는 의미를 정의하고 있다.

<정의 4> 불변조건은 빈 클래스가 항상 유지해야 하는 OCL 표현식으로 나타내어지는 조건들을 나타낸다.

불변조건은 *[규칙 5]*와 *[규칙 6]*에 따라 작성한다. *[규칙 5]*는 불변조건의 명세 형식을 규정하고 있으며, *[규칙 6]*은 불변조건을 표현하는 OCL 표현식의 작성 형식을 규정하고 있다.

[규칙 5] 빈 클래스의 불변조건은 다음과 같은 형태로 작성한다.

context *BeanClassName* **inv**:
OCL expression

[규칙 6] 불변조건, 사전조건, 사후조건을 구성하는 OCL 표현식(expression)은 OCL 명세서[17]의 사전 정의된 OCL 타입 및 문법에 따라 작성한다.

앞서 규정한 불변조건 작성 규칙(*[규칙 5], [규칙 6]*)의 적용 사례를 살펴보기 위해 도서 대출 처리 애플리케이션의 LibraryAccount 엔티티 빈을 고려해 보자. 표 3은 LibraryAccount 엔티티 빈이 유지해야 하는 제약조건을 나타내고 있다.

표 3 LibraryAccount 엔티티 빈의 제약조건

빈 이름	제 약 조 건
LibraryAccount	• 해당 계정에 대출된 도서의 양은 항상 양수이어야 함<1> • 해당 계정에 대출된 도서의 양은 항상 대출 한도량 이하이어야 함<2> • 대출된 모든 도서는 임대 불가능해야 함<3> • 대출된 모든 도서는 도서관에 존재해야 함<4>

명세 1은 *[규칙 5], [규칙 6]*을 적용한 Library Account 엔티티 빈이 유지해야 하는 불변조건 명세를 보여준다.

명세 1 LibraryAccount 엔티티 빈의 불변조건 명세

Context: *LibraryAccountBean*
Invariant:
self.lended->size >= 0 ····· <1>
self.lended->size <= self.lendingLimit ····· <2>
self.lended->forAll(b: Book | not b.BookBean.available) ····· <3>
self.lended->forAll(b: Book | self.library.book->exist(c:Book | b=c)) · <4>

위 명세에서 점선 이후의 표현은 OCL 표기법에는 없는 내용으로 표 3의 내용과 그에 대한 명세를 대조하기 위하여 임으로 삽입한 내용이다.

3.5 사전조건 및 사후조건 작성 규칙

사전조건 및 사후조건 또한 EJB 컴포넌트 클래스 다이어그램을 기반으로 일반적인 OCL 작성 규칙에 따라 작성한다. 따라서 가정은 필요하지 않다. 다음은 EJB 기반 애플리케이션에서 사전조건 및 사후조건이 가지는 의미와 빈 클래스의 오퍼레이션 종류를 정의하고 있다.

<정의 5> 사전조건은 빈 클래스의 각 오퍼레이션이 수행되기 위해 갖추어야 할 조건들을 나타낸다.

<정의 6> 사후조건은 사전조건이 만족될 때 빈 클래스의 각 오퍼레이션이 제공해 주어야 하는 조건들을 나타낸다.

<정의 7> 빈 클래스의 오퍼레이션은 일반적인 비즈니스 오퍼레이션과 애플리케이션 서버의 컨테이너에 의해 호출되는 EJB 필수(EJB Required) 오퍼레이션으로 구성된다.

사전조건 및 사후조건은 *[규칙 7]*과 *[규칙 6]*에 따라 작성한다. *[규칙 7]*은 사전조건 및 사후조건의 명세 형식을 규정하고 있으며, *[규칙 6]*은 사전조건 및 사후조건을 표현하는 OCL 표현식의 작성 형식을 규정하고 있다.

[규칙 7] 빈 클래스의 오퍼레이션에 대한 사전조건 사후조건은 다음과 같은 형태로 작성한다.

context *BeanClassName::operationName(param1 : Type1, ...): ReturnType*

pre *{precondition name} : OCL expression*

post *{postcondition name} : OCL expression*

[규칙 6] 불변조건, 사전조건, 사후조건을 구성하는 OCL 표현식(expression)은 OCL 명세서[17]의 사전 정의된 OCL 타입 및 문법에 따라 작성한다.

앞서 규정한 불변조건 작성 규칙(*[규칙 7], [규칙 6]*)의 적용 사례를 살펴보기 위해 도서 대출 처리 애플리케이션의 일반 비즈니스 오퍼레이션인 LendingClerk 세션 빈의 lending 오퍼레이션과 lendingExtension 오퍼레이션 그리고 EJB 필수 오퍼레이션인 Library-Account 엔티티 빈의 ejbCreate 오퍼레이션과 ejbRemove 오퍼레이션에 대하여 살펴본다. 표 4는 이들 오퍼레이션의 책임 및 권한을 나타내고 있다.

명세 2는 *[규칙 7], [규칙 6]*을 적용한 lending, lendingExtension, ejbCreate, ejbRemove 오퍼레이션의 사전조건 및 사후조건 명세를 보여준다.

위 명세에서 점선 이후의 표현은 OCL 표기법에는 없는 내용으로 표 4의 내용과 그에 대한 명세를 대조하기 위해 임으로 삽입한 내용이다.

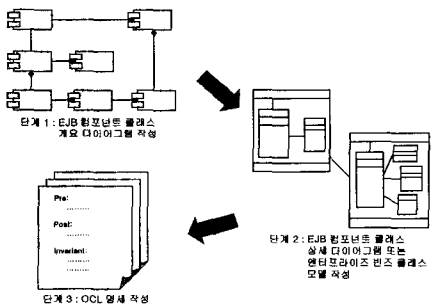


그림 8 EJB 기반 애플리케이션의 OCL 명세 작성 단계

표 4 lending, lendingExtension, ejbCreate, ejbRemove 오퍼레이션의 책임 및 권한

오퍼레이션 이름	책 임	권 한
lending	• 대출하고자 하는 도서는 도서관에 존재함(1) • 대출하고자 하는 도서는 대출 처리됨(2) • 해당 계정에 대출해간 도서의 목록을 추가시킴(3) • 해당 계정의 도서 대출 부수가 1 증가됨(4) • 대출해간 도서의 총 대출인 수가 1 증가됨(5)	• 대출하고자 하는 도서가 도서관에 존재해야 함(1) • 대출하고자 하는 도서는 이용 가능해야 함(2) • 해당 계정의 도서 대출 부수가 최대 대출 한계를 초과해서는 안됨(3) • 대출하고자 하는 도서가 자신이 대출중인 도서이면 안됨(4)
lendingExtension	• 대출을 연장한 도서는 이용 불가능한 상태가 됨(6) • 대출을 연장한 도서는 연장 처리됨(7) • 대출을 연장한 계정의 도서 대출량은 변함 없음(8) • 대출을 연장한 도서의 대출인 수는 변함 없음(9)	• 대출을 연장할 도서는 이미 본인에 의해 대여중이므로 이용 불가능해야 함(5) • 대출을 연장할 도서는 대출을 연장하고자 하는 계정에서 대여중이어야 함(6) • 대출을 연장할 도서는 예약되어 있지 않아야 함(7)
ejbCreate	• 새로 생성할 계정이 데이터베이스 내에 존재함(10) • 결과 값으로 새로 생성한 계정의 프라이머리 키 값을 반환함(11)	• 새로 생성할 계정은 사전에 존재하지 않는 계정이어야 함(8)
ejbRemove	• 삭제한 계정은 더 이상 데이터베이스 내에 존재하지 않음(12)	• 삭제할 계정은 데이터베이스 반드시 하나 존재해야 함(9)

3.6 명세 단계

본 논문에서는 *[규칙 1]* ~ *[규칙 7]*을 기반으로 그림 8에서 보이고 있는 명세 단계에 따라 EJB 기반 애플리케이션을 명세할 것을 제안한다. 명세 단계는 세 단계로 진행한다.

먼저 단계 1에서는 전체적인 애플리케이션의 구성을 모델링하기 위해 EJB 컴포넌트들간의 연관성과 다중성만을 고려한 EJB 컴포넌트 클래스 개요 다이어그램을 작성한다.

단계 2에서는 각각의 EJB 컴포넌트를 상세하게 모델링한다. 단계 1에서 작성한 EJB 컴포넌트 클래스 개요 다이어그램은 OCL 명세를 작성하기 위한 그래픽 모델로는 추상화 수준이 너무 높기 때문에 이를 구체화시키는 과정이 필요하다. 따라서 단계 2에서는 단계 1의 다이어그램을 구성하는 EJB 컴포넌트 각각에 대한 세부 모델인 EJB 컴포넌트 클래스 모델을 작성하고, 필요에

명세 2 lending, lendingExtension, ejbCreate, ejbRemove 오퍼레이션의 OCL 명세

Context: *LendingClerkBean::lending(aBook:Book)*
Pre :
self.LibraryAccount.library.book->exists(b:Book | b.bookBean.title = aBook.bookBean.title) ····· *(1)*
aBook.bookBean.available (2)
self.libraryAccount.lended->size < self.libraryAccount.libraryAccountBean.lendingLimit ····· *(3)*
self.libraryAccount.lended->count(aBook) = 0 ····· *(4)*
Post:
self.libraryAccount.library.book->exists(b:Book | b.bookBean.title = aBook.bookBean.title) ····· (1)
not aBook.bookBean.available (2)
self.libraryAccount.lended->count(aBook) = 1 ····· (3)
self.libraryAccount.libraryAccountBean.accountBalance
= self.libraryAccount.libraryAccountBean.accountBalance@pre + 1 ····· (4)
aBook.totalLendingCnt = aBook.totalLendingCnt@pre + 1 ····· (5)

Context: *LendingClerkBean::lendingExtension(aBook:Book)*
Pre :
not aBook.bookBean.available (5)
self.libraryAccount.lended->count(aBook) = 1 ····· *(6)*
not aBook.bookBean..reserved (7)
Post:
not aBook.bookBean.available (6)
self.libraryAccount.lended->count(aBook) = 1 ····· (7)
self.libraryAccount.libraryAccountBean.accountBalance
= self.libraryAccount.libraryAccountBean.accountBalance@pre ····· (8)
aBook.totalLendingCnt = aBook.totalLendingCnt@pre ····· (9)

Context *LibraryAccountBean::ejbCreate(paccountID : Integer, pownerName : String,*
plendingLimint : Integer) : LibraryAccountPK
Pre :
self.libraryAccountTB->select(accountID = paccountID)->isEmpty ····· *(8)*
Post:
self.libraryAccountTB->select(accountID = paccountID and ownerName = pownerName
and lendingLimit = plendingLimit and accountBalance = 0)->size = 1 ····· (10)
result = self.libraryAccountPK.LibraryAccountPK(paccountID) ····· (11)

Context *LibraryAccountBean::ejbRemove()*
Pre :
self.libraryAccountTB->select(id1:libraryAccountTB |
self.libraryAccountPK->select(id2:libraryAccountPK |
id1.accountID = id2.accountID))->size = 1 ····· *(9)*
Post:
self.libraryAccountTB->forAll(id1:libraryAccountTB |
self.libraryAccountPK->forAll(id2:libraryAccountPK |
id1.accountID <> id2.accountID)) ····· (12)

따라 EJB 컴포넌트 클래스 개요 다이어그램의 일부를 EJB 컴포넌트 클래스 모델로 연관성과 다중성을 표현한 EJB 컴포넌트 클래스 상세 다이어그램을 작성한다. EJB 컴포넌트 클래스 상세 다이어그램은 단계 1에서의 EJB 컴포넌트 클래스 개요 다이어그램과 별개의 다이어그램이 아니라 EJB 컴포넌트 클래스 개요 다이어그램의 일부를 상세하게 표현한 다이어그램이다.

단계 1과 단계 2에서 작성한 다이어그램은 OCL 명세를 작성하기 위해 필요한 그래픽 모델이다. 단계 3에서는 이러한 그래픽 모델을 기반으로 OCL 명세를 작성한다.

3.7 기존의 DbC 이용 명세 기법과의 차이점

DbC 접근법의 적용은 기존에 객체 지향 방법론에 주로 사용되어 왔으며, 본 논문에서 명세 언어로 사용한 OCL 언어는 DbC 접근법의 지원을 위해 객체지향 모델링 기법인 UML에 포함된 명세 언어이다[12]. 따라서 UML의 OCL 언어를 통하여 DbC 접근법을 이용한 객체 지향 애플리케이션의 명세 기법과 이러한 기법의 소프트웨어 공학적 장점들은 많은 연구들에서 이미 소개되었다. 그러나 EJB 기반 애플리케이션은 일반 객체지향 애플리케이션과는 다른 CBD 방법론이라는 큰 차이점이 존재하며, EJB 고유의 아키텍처를 가진다. 따라서 본 논문에서는 EJB 기반 애플리케이션을 명세할 수 있도록 몇 가지 새로운 기법을 제안하였다.

먼저 EJB 기반 애플리케이션을 일반적인 UML 클래스 다이어그램 형태로는 표현할 수 없기 때문에 클래스 다이어그램을 대체할 수 있는 새로운 다이어그램 규칙([규칙 1]~[규칙 4])을 제안하였다. 클래스 다이어그램은 OCL 언어를 이용하기 위한 기반이 되는 그래픽 모델로 DbC 접근법을 이용한 OCL 명세를 작성하기 위해서는 꼭 필요한 그래픽 모델이다.

또한 사전조건, 사후조건, 불변조건의 컨텍스트를 정의하였다. 사전조건은 오퍼레이션의 사용자가 지켜야할 의무를 나타내는 제약조건 기술 부분이며, 사후조건은 서비스의 공급자가 지켜야할 의무를 나타내는 제약조건이고, 불변조건은 항상 유지되어야 하는 제약조건을 의미한다. 기존의 객체지향 애플리케이션 명세에서 사전조건과 사후조건의 컨텍스트는 클래스의 각 오퍼레이션이 되며, 불변조건의 컨텍스트는 클래스 자신이 된다. 그러나 EJB 기반 애플리케이션은 일반적인 객체지향 애플리케이션과는 다른 아키텍처를 가지기 때문에 컨텍스트를 재정의할 필요가 있다. 따라서 본 논문에서는 사전조건 및 사후조건의 컨텍스트는 빈 클래스의 각 오퍼레이션으로하고 불변조건의 컨텍스트는 빈 클래스로 할 것을 제안하였다.

이러한 기법들을 토대로 본 논문에서는 소프트웨어 공학적 관점에서 많은 장점들을 지닌 DbC 접근법을 EJB 기반 애플리케이션의 명세에 이용하기 위한 기법을 제안하고 있으며, 이러한 사항은 기존의 DbC와 관련된 연구들이 객체지향 방법론 측면의 연구였다는 점에서 가장 큰 차이점으로 볼 수 있다. 따라서 본 논문에서 제안한 명세 기법은 DbC 접근법 자체가 지니고 있는 '소프트웨어의 신뢰성 향상'이라는 장점을 CBD 방법론이 가질 수 있도록 해 준다.

4. 명세의 의의

EJB 기반 애플리케이션 구축에는 그림 9에서와 같이 빈 제공자, 애플리케이션 조립자, 배치자, EJB 컨테이너/서버 제공자, 시스템 관리자가 참여한다[9]. 빈 제공자는 EJB 기반 애플리케이션을 구성하는 각각의 엔터프라이즈 빈즈를 생성하고, 애플리케이션 조립자는 빈 제공자가 생성한 엔터프라이즈 빈즈들을 조립하여 애플리케이션을 구축한다. 배치자는 EJB 컨테이너/서버 제공자가 제공하는 애플리케이션 서버에 애플리케이션 조립자가 구축한 애플리케이션을 배치하고, 배치된 애플리케이션은 시스템 관리자에 의해 유지보수 된다. 본 논문에서 기술한 명세는 이들 참여자들에게 다음 세 가지 의미로 응용될 수 있으며, 그림 9는 DbC 접근법을 이용한 EJB 기반 애플리케이션의 OCL 명세가 각각의 참여자들에게 가지는 의미를 보여준다.

- *소프트웨어 설계 방법으로서의 의미*
- *소프트웨어 문서로서의 의미*
- *소프트웨어의 테스팅 사례 추출을 위한 근거로서의 의미*

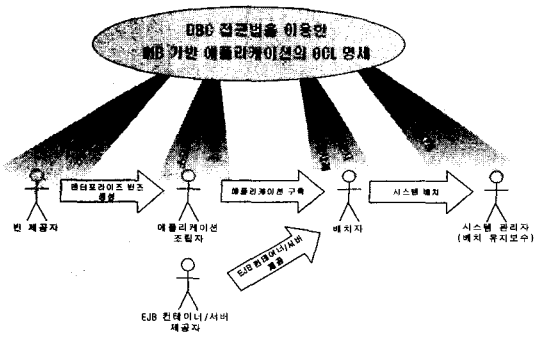


그림 9 DbC 접근법을 이용한 EJB 기반 애플리케이션의 OCL 명세가 가지는 의미

소프트웨어의 설계 방법으로서 의미는 앞 절에서 기술하였던 명세를 빈 제공자가 엔터프라이즈 빈즈를 개발하기 위한 설계 방법으로 채택할 수 있다는 의미이다. 이러한 소프트웨어 컨트랙트 개념은 상호 책임과 권리를 명확히 정의함으로써 모호함이 없이 소프트웨어를 설계할 수 있도록 해 준다[14]. 따라서 앞서 제안한 명세 방법의 기반이 되는 DbC 접근법은 설계의 관점에서 보면 견고한 소프트웨어 개발의 기반이 된다.

소프트웨어의 문서로서의 의미는 앞 절에서 기술하였던 명세를 애플리케이션 조립자, 배치자, 시스템 관리자가 각각의 엔터프라이즈 빈즈 및 애플리케이션을 정확히 이해하기 위한 문서로서 참조할 수 있다는 의미이다. 즉, 애플리케이션 조립자는 빈을 조립하기 위한 정보로서, 배치자는 조립자가 구축한 애플리케이션을 애플리케이션 서버에 배치하기 위한 정보로서, 시스템 관리자는 시스템을 유지보수하기 위한 정보로서 명세를 활용할 수 있다는 것이다.

소프트웨어의 테스팅 사례 추출을 위한 근거로서의 의미는 앞 절에서 기술하였던 명세가 빈 제공자, 애플리케이션 조립자, 배치자, 시스템 관리자가 수행해야 하는 각각의 테스팅에 필요한 테스트 사례를 제공해 준다는 의미이다. 최근 EJB 컴포넌트의 테스팅을 자동화하기 위해 DbC 접근법의 적용이 시도되고 있다[18]. 즉, DbC 접근법의 제약사항 개념을 이용하여 이를 테스팅의 자동화를 위한 테스팅 사례로서 활용하는 것이다. 그러나 현재는 오퍼레이션의 파라미터나 반환하는 결과값이 유지해야 할 제약조건만을 테스팅 사례로 활용하고 있는 실정이다. 그 이유는 EJB 컴포넌트에 DbC 접근법을 적용할 명세 규칙이 명확하지 않기 때문이다. 따라서 본 논문에서 제안한 DbC 기법을 적용한 EJB 기반 애플리케이션의 명세 기법은 보다 완벽한 테스팅 사례를 추출하는데 충분한 활용 가치를 가지고 있다.

5. 결론 및 향후 연구

CBD는 요구사항 분석, 설계, 구현, 테스트, 배치 및 기타 기술적인 지원 등을 포함하는 개발 라이프사이클의 모든 측면 및 단계를 컴포넌트에 기반을 두고 있는 소프트웨어 개발 접근법으로 현재 꾸준히 소프트웨어 개발 분야에서 성장해 가고 있다. 이에 편승하여 CBD 방법론의 한 가지로서 분산 웹 애플리케이션 구축 시 복잡한 분산 프레임워크와 관련된 코드의 작성 없이 비즈니스 코드 작성에 주력할 수 있도록 해 주는 EJB 역시 소프트웨어 개발에 많이 적용되고 있다. 그러나 많은 사용에도 불구하고 EJB 기반 소프트웨어의 신뢰성 문제에 대해서는 많은 노력이 기울여지고 있지 않는 실정이다. 따라서 본 논문에서는 신뢰성 있는 EJB 기반 애플리케이션 개발이라는 목적을 달성하기 위해 기존의 객체지향 방법론에 적용되어 소프트웨어의 신뢰성 면에서 많은 이점을 가져온 DbC 접근법[19]을 이용한 명세 기법을 제안하였다.

명세 기법의 제안을 위해 본 연구에서는 다음과 같은 작업을 수행하였다. 먼저 DbC 접근법을 적용하여 정형 명세를 만들어내기 위해 DbC 접근법을 지원해 주는 명세 언어를 선정하였다. 명세 언어로서는 많은 수학적 배경지식이 필요치 않으며, 인지도가 높은 UML의 OCL 언어를 선택하였다. 다음에는 선정한 OCL 언어의 특징 및 용법을 분석하여 객체지향 애플리케이션이 아닌 EJB 기반 애플리케이션을 명세하는데 필요한 사항들을 식별하였다. 이 과정에서 OCL 언어를 이용하여 EJB 기반 애플리케이션을 명세하기 위해서는 UML의 클래스 다이어그램과 같은 형태의 그래픽 모델이 필요함을 인식하였고, 따라서 먼저 명세의 기반이 되는 그래픽 모델과 엔터프라이즈 빈즈를 구성하는 구성 요소 중에서의 명세 대상과 관련된 기초 가정((가정 1)~(가정 2))을 설정하고, 그래픽 모델을 구성하는 구성요소들에 대한 가정((가정 3)~(가정 5))과 구성요소들간의 다중성, 연관성의 정의(<정의 1>~<정의 3>)를 통해 EJB 컴포넌트 클래스 다이어그램 작성 규칙([규칙 1]~[규칙 4])을 도출하였다. 마지막으로는 불변조건, 사전조건, 사후조건이 EJB 기반 애플리케이션에서 가지는 의미를 정의(<정의 4>~<정의 7>)하고, 이러한 정의를 통해 EJB 컴포넌트 클래스 다이어그램을 기반으로 사전조건, 사후조건, 불변조건을 OCL 표현식으로 표현하기 위한 표현 규칙([규칙 5]~[규칙 7])을 도출하였다. 이러한 가정, 정의, 규칙은 EJB 명세서의 EJB 아키텍처, DbC 접근법의 사전조건, 사후조건, 불변조건의 의미, UML의 클래스 다이어그램 표기법 및 OCL 문법에 준하여 제시한 기법이므로 이러한 기법을 통해 어떠한 EJB 기반 애플리케이션의 명세도 가능하다. 이와 같은 과정을 통해 얻어낸 명세 기법은 다음 세 가지 의미를 지닌다.

첫째, 소프트웨어의 설계 방법으로서 의미
둘째, 소프트웨어의 문서로서의 의미
셋째, 소프트웨어의 테스팅 사례 추출을 위한 근거로서 의미

Beugnard 등은 컨트랙트를 네 가지 레벨(기본 컨트랙트, 행위적 컨트랙트, 동기화 컨트랙트, QoS)로 분류하여 정의하였다. 본 논문에서 명세한 명세 수준은 그들이 정의한 네 가지 레벨 중 하위 두 레벨만을 다룰 수

있는 명세이다. 따라서 나머지 두 가지 레벨을 다룰 수 있는 명세 방법에 관한 지속적인 연구가 필요하다.

또한 명세는 'How' 문제가 아닌 'What'의 문제이다. 즉, 애플리케이션의 구현 방법을 명세하는 것이 아니라 애플리케이션이 어떤 기능을 수행하는지를 명세하는 것이다. 그러나 EJB는 반드시 따라야할 그 자신만의 아키텍처를 가지며, 따라서 EJB 기반 애플리케이션을 명세할 때 EJB의 아키텍처를 고려한다면 어느 정도 구현 방법이 명세에 포함되는 것은 불가피하다. EJB 특유의 아키텍처에 따라 명세에 포함되는 How 문제 즉, 구현 방법의 문제의 명세 범위에 관한 연구도 향후 연구 과제로 요구되며, 또한 본 연구의 명세 기법은 Assertion 측면의 CBD 명세 기법이다. 이러한 CBD 명세 기법에서 고려할 수 있는 측면은 이 외에도 동기화(synchronization), 이벤트(event), 외부 기능적/행위적 측면들을 고려할 수 있다. 따라서 이러한 측면을 고려한 명세 기법의 접근 또한 요구된다.

참 고 문 헌

[1] Peter Hersum, Oliver Sims, *Business Component Factory*, Wiley, 2000.

[2] B. Meyer, "Applying 'design by contract'", Computer, pp. 40-51, October, 1992.

[3] B. Baudry, Vu Le Hanh, Y. Le Traon, "Testing-for-Trust:the Genetic Selection Model Applied to Component Qualification", Technology of Object-Oriented Languages, TOOLS 33, Proceedings pp. 108-119, 2000.

[4] K. Arnout, r. Simon, "The .NET Contract Wizard: Adding Design By Contract to Language Other Than Eiffel", Technology of Object-Oriented Languages and Systems, TOOLS 39 International Conference and Exhibition, pp. 14-23, 2001.

[5] ISE, "Building bug-free O-O software : An introduction to Design By Contract", http://www.eiffel.com/doc/manuals/technoloty/contract/page.html, 2000.

[6] Jean-Marc Jézéquel, and Bertrand Meyer, "Design By Contract: The Lessons of Ariane", *Computer*, pp. 129-130, Jan, 1997.

[7] R. Kramer, "iContract-The Java™ Design by Contract™ Tool", Technology of Object-Oriented Languages, TOOLS 26 Proceedings, pp. 295-307, 1998.

[8] Yu Liu, "From UML to Design By Contracts", JOOP, April, 2001.

[9] Ed Roman, *Mastering Enterprise JavaBeans*, Wiley, 2000.

[10] B. Meyer, "Design by Contract: Making Object-Oriented Programs that Work", Technology of Object-Oriented Languages and Systems, TOOLS 25 Proceedings, pp. 360-361, 1998.

[11] R. Plosch, "Technology of Object-Oriented Languages and Systems", TOOLS 26 Proceedings, pp. 282-294, 1998.

[12] Warmer, Kleppe, *The Object Constraint Language*, Addison Wesley, 1998.

[13] B. Meyer, "Design By Contract: The Eiffel Method", Technology of Object-Oriented Languages, TOOLS 26, Proceedings, pp. 446-446, 1998.

[14] Benoit Baudry, Yves Le Traon, Jean-Marc Jézéquel, "Robustness and Diagnosability of OO Systems Designed By Contracts", Software Metrics Symposium, pp. 272-284, 2001.

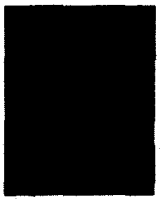
[15] 노혜민, 이상영, 김송주, 유철중, 장옥배, 이우진, 신규상, "서버측 애플리케이션 개발을 위한 EJB 지원 엔터프라이즈 빈즈 생성기/전개기의 설계 및 구현", 한국정보과학회 논문지:컴퓨팅의 실제, pp. 429-439, 2001.

[16] 노혜민, 유철중, 장옥배, "DBC 접근법을 이용한 엔터프라이즈 빈즈 명세 기법", 한국정보과학회 추계 학술발표 논문집(I) 제28권 2호, pp. 421-423, 2001.

[17] OMG, OMG Unified Modeling Language Specification, http://www.rational.co.kr/leadership/uml_resource_center.asp, 1999.

[18] ParaSoft, *Using Design By Contract™ to Automate Java™ Software and Component Testing*, 2001.

[19] Richard Mitchell, Jim McKim, *Design by Contract, by Example*, Addison Wesley, 2002.



노 혜 민

2000년 전북대학교 컴퓨터과학과 졸업(이학사). 2002년 전북대학교 대학원 전산통계학과 졸업(이학석사). 2002년~현재 전북대학교 대학원 컴퓨터통계정보학과 박사과정. 관심 분야는 컴포넌트 기반 소프트웨어 개발, 정형 명세 기법, 소프트웨어공학 등

유 철 중

1982년 전북대학교 전산통계학과 졸업(이학사). 1985년 전남대학교 대학원 계산통계학과 졸업(이학석사). 1994년 전북대학교 대학원 전산통계학과 졸업(이학박사). 1982년~1985년 전북대학교 전자계산소 조교. 1985년~1996년 전주기전여자대학 전자계산과 부교수. 1997년~현재 전북대학교 자연과학대학 컴퓨터과학과 조교수. 관심분야는 소프트웨어공학, 에이전트공학, 컴포넌트기술, 분산객체기술, GNSS(GPS), GIS, 멀티미디어, 인지과학 등