# J2EE 플랫폼에서의 개념적 컴포넌트 모델링 및 컴포넌트 생성 지원 도구 개발

이 우 진[†]·김 민 정[††]·정 양 재[††]·윤 석 진[†]
최 연 준[††]·이 지 현[††]·신 규 상[†††]

## 요 약

소프트웨어 산업이 급속하게 발전해감에 따라 정보 기술 업체간 경쟁이 더욱 심화되어 소프트웨어 재사용성, 적시성, 유지 보수성 등이 업체의 생명력으로 대두되면서 소프트웨어 컴포넌트 기술이 점차 각광을 받고 있다. 현재 몇몇 컴포넌트 생성 지원 도구들이 제공되고 있지만 컴포넌트의 식별, 모델링, 상세 설계, 코드 생성, 전개, 시험 등 컴포넌트 생성 전 과정을 밀접히 연계하여 지원하는 도구가 드물다. 또한, 특정 플랫폼에 의존적인 소규모 컴포넌트 생성에 중점을 두고 있어 사용자 관점의 다양한 규모의 컴포넌트 생성에는 제약이 따른다. 이 논문에서는 컴포넌트 생성에 연관된 모든 과정을 지원하는 컴포넌트 모델링 및 생성 지원 도구의 설계와 프로토타입 구현에 대해 기술한다. 컴포넌트 모델링은 영역 고유의 비즈니스 로직의 재사용 측면에서 컴포넌트 플랫폼 아키텍처에 관계없이 개념적인 컴포넌트의 식별 및 모델링을 지원한다. 상세설계 및 코드 생성 부분은 일차적으로 J2EE 플랫폼 아키텍처에 의존적으로 지원되며 설계 모델과 소스 코드의 일관성을 동적으로 유지시키는 Round-trip Engineering 기능을 지원한다.

# Development of a Supporting Tool for Conceptual Component Modeling and Component Construction on the J2EE Platform

Woo Jin Lee[†]·Min Jeong Kim[††]·Yang Jae Jeong[††]·Seok Jin Yoon[†]
Yeon Jun Choi[††]·Ji Hyun Lee[††]·Gyu Sang Shin[†††]

## ABSTRACT

As software industry is rapidly evolving, IT business enterprises have been meeting with cutthroat competition in developing software. As software reusability, time to market, and maintainability are considered as a competitive edge, software component techniques have lately attracted considerable attention. Currently, although there are some supporting tools for developing software components, they do not have tight connections among component developing processes such as component identification, component modeling, detailed design, code generation, deployment, and testing. And it is restrictive for users to construct various scales of components on component platform architecture. In this paper, we provide an implementation and a design of a supporting tool for constructing platform-independent software components, which covers all development lifecycles of components. In the phase of component modeling, platform independent, conceptual components are identified from domain model information in the view of system partitioning. Detailed design and implementation of a component are performed on the J2EE platform architecture. And the changes on the design model and source codes are consistently managed by using round-trip techniques.

**키워드 : 컴포넌트(Component), 컴포넌트 생성 도구(Component Construction Tool), EJB, 컴포넌트 모델링(Component Modeling), J2EE, CBD**

## 1. 서 론

최근 들어, 정보 기술 산업의 빠른 성장으로 소프트웨어 적용 분야가 다양해지고 응용 소프트웨어 또한 점차 복잡해지고 있다. 이러한 시장 수요에 부응하기 위해서는 좀더 짧은 시간 내에 보다 손쉽게 소프트웨어를 개발할 수 있는 기술이 절실히 요구되고 있다. 하지만 구조적 방법론, 객체지향 방법론 등의 기존 개발 방법론만으로는 소프트웨어의 생산성을 향상시키는데 한계를 느끼기 시작하였다. 이러한 시점에서, 최근 업체로부터 점차 각광을 받으며 부상하고 있는 기술이 소프트웨어 컴포넌트 기술이다. 소프트웨어 컴포넌트 기술은 컴포넌트 단위로 독립적인 개발이 가능함으

† 정 회 원 : ETRI 컴퓨터소프트웨어연구소 S/W공학연구부 선임연구원
†† 정 회 원 : ETRI 컴퓨터소프트웨어연구소 S/W공학연구부 연구원
††† 정 회 원 : ETRI 컴퓨터소프트웨어연구소 S/W공학연구부 컴포넌트공학연구팀 팀장
논문접수 : 2001년 10월 5일, 심사완료 : 2001년 12월 15일

로 병행적, 단계적 소프트웨어 개발을 통한 개발 기간 단축 효과를 얻을 수 있다. 그리고 컴포넌트의 수행은 SUN의 Enterprise JavaBeans (EJB)[1], OMG의 CORBA Component Model(CCM)[2], 마이크로소프트의 DCOM[3] 등의 컴포넌트 플랫폼 아키텍쳐상에서 이루어지므로 플랫폼 아키텍쳐가 제공하는 트랜잭션, 보안성, 지속성 등의 다양한 서비스를 이용할 수 있으며 또한 컴포넌트 단위로 수행되므로 컴포넌트의 대체 및 유지보수가 수월한 장점이 있다.

컴포넌트 관련 개발 프로세스는 크게 컴포넌트를 조립하여 어플리케이션을 개발하는 조립 프로세스와 각각의 컴포넌트를 개발하는 생성 프로세스로 나뉘어 진다. 이 논문에서는 이중에서 컴포넌트 생성 프로세스에 중점을 둔다. 컴포넌트 생성 프로세스는 영역 모델링, 컴포넌트 식별 혹은 추출, 컴포넌트 모델링, 컴포넌트 상세 설계 및 구현, 컴포넌트 전개 및 시험 단계로 이루어진다.

지금까지 소개된 대표적인 컴포넌트 생성 도구로는 Select사의 SCF(Software Component Factory)[5], Computer Associates 사의 COOL : Joe [6], TogetherSoft 사의 Together[7] 등이 있다. <표 1>은 컴포넌트 생성 프로세스의 각 단계에서 이용할 수 있는 도구들의 기능을 보여주고 있다. <표 1>에서 '-'로 표시된 부분은 도구에서 언급되어 있지 않은 특성을 나타내고 '△'로 표시된 부분은 일부 기능만을 제공하는 경우이며 'O'는 완전한 기능을 제공하는 것을 나타낸다. Select의 SCF와 Together는 컴포넌트의 내부 설계 및 구현 기능에 대해서는 지원하지만 컴포넌트 식별 및 모델링 부분에 대한 지원이 부족하다. 반면, COOL : Joe의 경우는 EJB 컴포넌트에 특화된 컴포넌트 식별 및 모델링 기능이 지원되며 동적 Round-trip 지원에서도 순방향만 지원한다.

<표 1>에서 알 수 있듯이, 기존 도구들은 전체 생성 프로세스를 체계적이며 유기적으로 지원하지 못하고 특정 단계에 치중되어 있는 모습을 알 수 있다. 그리고 COOL : Joe와 같이 플랫폼 의존적인 컴포넌트 모델링을 지원하면 플랫폼에 의존적인 컴포넌트를 만들게 됨으로 추후에 다른 플랫폼인 CCM이나 COM/DCOM 환경에서 재사용하는데 문제가 있다. 플랫폼간에 재사용성을 높이기 위해서는 영역의 비즈니스 로직만을 고려하여 컴포넌트 모델링하고 이들 컴포넌트를 특정 플랫폼에 맞게 구현하는 2단계 접근 방법이 필요하다.

이 논문에서는 <표 1>에 나타난 영역 모델링, 컴포넌트의 식별 및 추출, 컴포넌트 모델링, 컴포넌트 상세설계 및 코드 생성, 배치 및 시험 등 컴포넌트 생성 프로세스 전과정을 지원하는 컴포넌트 생성 지원 도구의 설계 및 구현에 대해 다룬다. 그리고 컴포넌트 모델링에서는 영역 특성을 고려하여 재사용 가능한 비즈니스 로직별로 시스템을 분할하는 개념적 컴포넌트 모델링과 특정 플랫폼에 맞춘 내부

〈표 1〉 컴포넌트 관련 도구의 기능 비교

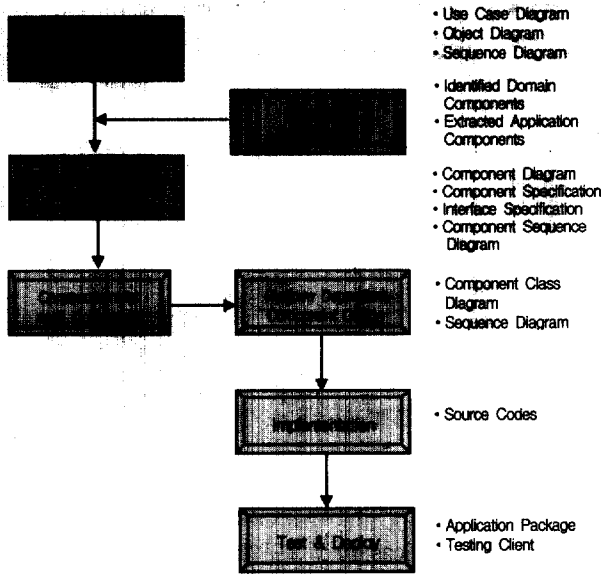
| | Select SCF | COOL:Joe 2.0 | Together 5.0 | COBALT: Constructor |
|---|---|---|---|---|
| 영역 모델링 지원 | • BP 모델링<br>• UC 모델링 | • UC Modeling<br>• Type Diagram<br>• 순차도 | UML 다이어그램 모델링 | • UC 모델링<br>• 객체모델링<br>• 순차도 |
| 컴포넌트 식별 기능 지원 | - | △ | - | O |
| 소스에서의 EJB 컴포넌트 추출 기능 | - | - | - | O |
| 컴포넌트 모델링 기능 (컴포넌트 다이어그램 지원 여부) | - | 플랫폼 의존적 | - | 플랫폼 독립적 |
| 컴포넌트 상세 설계 및 동적 Round-trip 기능 지원 | O | (순방향)<br>△ | O | O |
| 컴포넌트 배치 및 시험 (사용자 정의 객체 시험 가능 여부) | -<br>(-) | O<br>(-) | O<br>(-) | O<br>(O) |

컴포넌트 클래스 모델링으로 구분하여 영역 컴포넌트의 재사용성 및 플랫폼간의 이식성을 향상시킨다. 이 논문에서는 EJB 컴포넌트 모델을 우선적으로 고려하여 개념적 컴포넌트를 EJB 컴포넌트로 생성, 배치하는 과정을 지원한다. EJB 컴포넌트의 배치는 Java 2 Platform Enterprise Edition (J2EE) 표준에 맞게 컴포넌트를 묶고 어플리케이션 서버에 탑재하는 과정을 지원한다. 그리고 설계 다이어그램과 소스 코드와의 일관성을 동적으로 유지시키기 위해 SRE(Simultaneous Round-trip Engineering) 기능을 가지는 클래스 다이어그램 편집기 및 소스 코드 편집기를 제공한다.

이 논문의 구성은 다음과 같다. 먼저 제2절에서는 컴포넌트 생성 지원 도구의 개략적인 개발 프로세스와 본 논문에서 제안하고 있는 COBALT(COmponent Based Application deveLopment Tool) : Constructor 도구의 전체 블록 관계도를 설명한다. 제3절에서는 개념적인 컴포넌트 모델링에 대해 기술하며 제4절에서는 J2EE 환경에 맞춰진 컴포넌트 상세 설계 및 구현 과정에 대해 기술한다. 제5절에서는 EJB 컴포넌트의 배치 및 인터페이스 시험 환경에 대해 언급하고 마지막으로 제6절에서는 결론 및 향후 연구 방향에 대해 기술한다.

## 2. 컴포넌트 생성 지원 도구의 개략 설계

컴포넌트 기반 소프트웨어 개발 프로세스는 도메인 분석, 컴포넌트 모델링, 컴포넌트 설계, 컴포넌트 구현, 컴포넌트 배치 및 시험 과정으로 나뉘어진다. (그림 1)은 이러한 프로세스들을 나타내고 있으며 각 단계별로 필요한 모델링 산출물들을 오른쪽에 정리하여 보여주고 있다. 각 단계에 대한 개략적인 설명은 다음과 같다.




(그림 1) 컴포넌트 모델링 및 개발 프로세스

단계 1 : 마르미 III[9]에서 제안된 공통성 및 가변성 분석 방법을 통해 얻어진 어플리케이션 도메인의 영역 정보를 바탕으로 Use Case 모델, 객체 모델, 순차도(Sequence Diagram)를 생성한다.

단계 2 : 객체 모델에 기술된 객체들의 연관 관계 정보와 Use Case에 나타난 객체들의 기능적 연관성 및 사용 횟수 등을 고려하여 개념적 컴포넌트 식별이 이루어진다. 컴포넌트 식별 단계에서는 영역 모델링을 이용하여 개념적인 컴포넌트들을 식별할 뿐만 아니라 기존 자바 시스템을 분석하여 EJB 컴포넌트를 추출하는 과정도 포함한다.

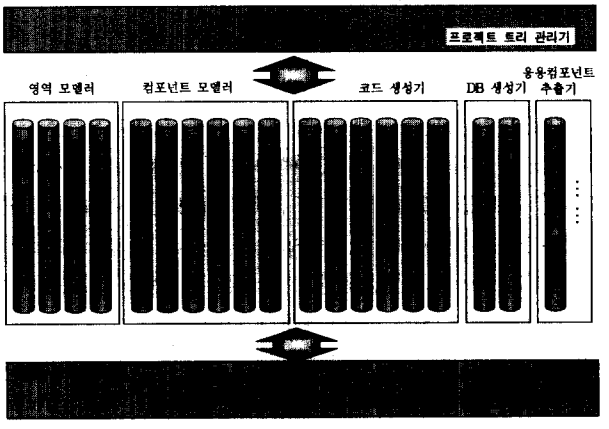
단계 3 : 컴포넌트 다이어그램을 통해 개념적 컴포넌트들에 대해서 컴포넌트간의 의존성 명시하고 각 컴포넌트의 인터페이스를 명확히 정의한다. 이때, 컴포넌트 인터페이스를 명확히 정의하기 위해 컴포넌트간의 메시지 흐름을 순차도로 작성할 수 있다.

단계 4 : 컴포넌트 인터페이스가 정의되면 각각의 컴포넌트별로 독립적으로 내부 설계 및 구현이 가능하다. 컴포넌트의 내부설계 및 구현은 컴포넌트 플랫폼 아키텍처의 특성을 고려하여야 하므로 컴포넌트 플랫폼에 의존적이다. 이 논문에서는 J2EE 플랫폼의 EJB 컴포넌트 개념을 우선적으로 지원한다. 향후 버전에서는 DCOM, CCM 등을 지원할 예정이다.

단계 5 : 컴포넌트의 구현이 완료되면 EJB 소스를 JAR로 묶어서 어플리케이션 서버에 배치하고 컴포넌트의 인터페이스를 간단히 시험할 수 있는 클라이언트 프로그램을 생성한다.

이러한 컴포넌트 생성 프로세스를 지원하기 위해 COBALT : Constructor 도구는 (그림 2)와 같이 영역 모델러, 컴포넌트 모델러, 코드 생성기, 응용 컴포넌트 추출기, 데이터베이스 연계기, 통합 GUI, 그리고 모델정보 관리기의 총 7개의 서브 시스템으로 구성된다. 각 서브 시스템에 대한 개략적인 설명은 다음과 같다.

- **영역 모델러** : 영역 모델러는 특정 영역에 속하는 여러 어플리케이션들의 공통 요소들을 추출하여 모델링하기 위해 Use Case 다이어그램, 객체 모델, 순차도를 작성할 수 있는 기능을 제공하며, 또한 이러한 정보를 바탕으로 후보 컴포넌트들을 식별하는 기능을 제공한다.
- **컴포넌트 모델러** : 컴포넌트 모델러는 각 컴포넌트의 범위를 정하고 각 컴포넌트의 인터페이스를 명확히 기술하며 또한 컴포넌트의 상세 설계를 위해 내부 클래스들을 정의할 수 있는 도구이다. UML(Unified Modeling Language)의 컴포넌트 다이어그램, 클래스 다이어그램, 순차도, 설계 패턴, OCL, XML 변환 기능이 제공되며, 다양한 다이어그램을 효율적으로 사용하거나 여러 다이어그램들 간의 원활한 산출물 공유를 위해 뷰와 모델을 분리하여 관리한다.
- **코드 생성기** : 코드 생성기는 컴포넌트 다이어그램의 각 컴포넌트 별로 내부 비즈니스 로직을 구현하고 컴포넌트들을 패키징하여 배치, 시험할 수 있는 환경을 제공한다. 그리고 상세 설계와 소스 코드의 일관성을 유지하기 위해 SRE 기능을 제공한다.
- **응용 컴포넌트 추출기** : 응용 컴포넌트 추출기는 기존의 자바 언어로된 프로그램으로부터 컴포넌트를 추출하여 컴포넌트 다이어그램을 생성하고 EJB 컴포넌트로 변환한다.




(그림 2) 컴포넌트 생성 지원 도구의 블록도

- **데이터베이스 연계기** : 데이터베이스 연계기는 데이터베이스의 테이블을 생성, 관리할 수 있는 환경을 제공한다. 사용자는 데이터베이스 연계기를 통해 쉽게 DB 테이블을 생성할 수 있고 컴포넌트 설계 시에는 DB 테이블로부터 손쉽게 EJB의 엔티티 빈(entity bean)을 생성할 수 있는 기능을 제공한다.

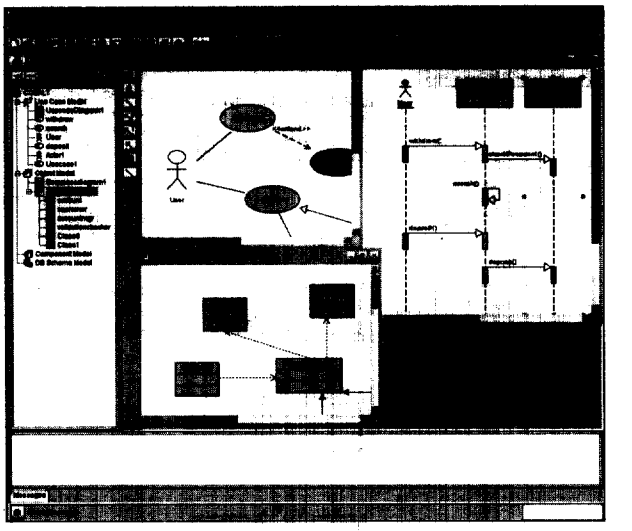
- **통합 GUI** : 통합 GUI는 쉽게 도구를 이용할 수 있도록 사용자 중심의 GUI 인터페이스를 제공하며 여러 다이어그램 편집기들을 체계적으로 관리하여 원활히 연계하여 동작할 수 있게 하며 브라우징 윈도우를 통해 모델링 산출물을 시각적으로 관리하는 기능을 제공한다.
- **모델 정보 관리기** : 모델 정보 관리기는 Use Case 모델, 객체모델을 관리하는 도메인 모델 관리기, DB 스키마 모델을 관리하는 스키마 모델 관리기, 컴포넌트 다이어그램과 각 컴포넌트를 관리하는 컴포넌트 모델 관리기, 컴포넌트 전개기에 의해 만들어진 전개 파일들을 관리하는 전개 모델 관리기로 구성된다.

## 3. 개념적인 컴포넌트 모델링

개념적인 컴포넌트는 독립적으로 수행될 수 있는, 특정 영역의 비즈니스 로직만을 고려하여 구성된 재사용 단위로 J2EE, DCOM, CCM 등의 플랫폼 기술과 무관하다. 이러한 개념적인 컴포넌트들이 실제 플랫폼에서 수행되기 위해서는 반드시 플랫폼 기술을 고려하여 상세 설계 및 구현이 이루어져야 한다. COBALT : Constructor 도구에서 개념적 컴포넌트 모델링을 강조하는 이유는 일반적으로 설계 모델이 프로그램 언어에 독립적이듯이 구현 및 런타임 특성인 플랫폼 기술에 독립적인, 비즈니스 로직을 재사용할 수 있는 컴포넌트 모델이 존재하여야 한다고 보기 때문이다. 그리고 개념적인 컴포넌트는 플랫폼 의존적인 소규모 컴포넌트에 비해 다양한 형태로 표현될 수 있으므로 독립적, 병행적 작업을 가능케 하는 시스템 분할의 의미도 내포되어 있다.

### 3.1 UML을 이용한 영역 모델링

COBALT : Constructor 도구에서 영역 모델링은 UML의 Use Case 다이어그램, 객체 다이어그램, 순차도를 이용하여 도메인 정보를 모델링한다. 또는 응용컴포넌트추출기의 역공학 기능을 이용하여 기존 자바 시스템의 소스 코드로부터 클래스 다이어그램, 순차도 정보를 추출하여 사용할 수도 있다. (그림 3)은 COBALT : Constructor 도구의 통합 GUI 및 Use Case 다이어그램, 객체 다이어그램 및 순차도를 작성하는 과정을 보여주고 있다. COBALT : Constructor 도구의 통합 GUI는 위쪽에 도구의 풀다운 메뉴 및 메뉴 아이콘들이 존재하며 왼쪽 창에는 모델링 산출물들을 관리하는 브라우징 윈도우가 있다. 그리고 오른쪽 편집 창에서 여러 다이어그램 편집기들을 이용할 수 있으며 아래 메시지 창에는 오류 메시지 혹은 컴파일 결과 등이 나타난다. (그림 3)의 편집창에서는 COBALT : Constructor 도구의 Use Case 다이어그램 편집기, 객체 다이어그램 편집기, 순차도 편집기의 실행 예를 보여주고 있다.



(그림 3) COBALT : Constructor 도구의 GUI 및 영역 모델링 정보

### 3.2 개념적 컴포넌트 식별 및 추출

개념적 컴포넌트의 식별은 도메인 전문 지식과 경험이 풍부한 전문가가 직접 식별하는 경우, 영역 모델링 정보를 바탕으로 식별 알고리즘을 수행하여 얻는 경우, 영역의 기존 시스템들을 분석하여 후보 컴포넌트를 추출하는 경우 등으로 크게 나뉘어 볼 수 있다. 첫 번째 경우는 도메인 전문가가 COBALT : Constructor 도구를 이용하여 영역 모델링을 거치지 않고 곧바로 컴포넌트 다이어그램을 작성할 수 있으며 나머지 두 경우는 COBALT : Constructor 도구의 영역 컴포넌트 식별기와 응용 컴포넌트 추출기를 이용하여 식별 과정을 수행할 수 있다.

영역 컴포넌트 식별기는 영역 모델링 정보로부터 객체 리스트, Use Case 정보, Use Case 별 객체사용 정보를 얻어 객체 의존성 네트워크를 생성하고 이를 바탕으로 객체 묶음 알고리즘을 수행하여 컴포넌트로 나뉘어질 수 있는 객체들을 그룹화 한다. 이때 객체 사용 정보는 순차도로부터 유추되는 것을 가정하고 있으며 만약 순차도 정보가 없을 경우는 사용자로부터 입력받는다. 생성된 컴포넌트 정보 및 컴포넌트 클래스 정보는 컴포넌트 다이어그램의 초기값으로 이용된다.

응용 컴포넌트 추출기는 기존의 자바 언어로 된 프로그램으로부터 컴포넌트를 식별하여 컴포넌트 다이어그램을 생성하고 식별된 컴포넌트를 EJB로 생성한다. 컴포넌트를 식별하기 위해 자바 코드 분석기가 소스코드를 분석하여 클래스 다이어그램을 생성하고, 생성된 클래스 다이어그램과 기타 분석 정보들을 바탕으로 후보 컴포넌트를 식별한다. 식별된 컴포넌트는 EJB 형식에 맞추어서 컴포넌트로 변환되고, 컴포넌트 모델러에서 사용될 수 있도록 컴포넌트 다이어그램 및 인터페이스 정의 형식으로 만들어져 컴포넌트 모델러에 전달된다. 응용 컴포넌트 추출기의 추출 알고리즘에 대한 자세한 내용은 [8]에 설명되어 있다.

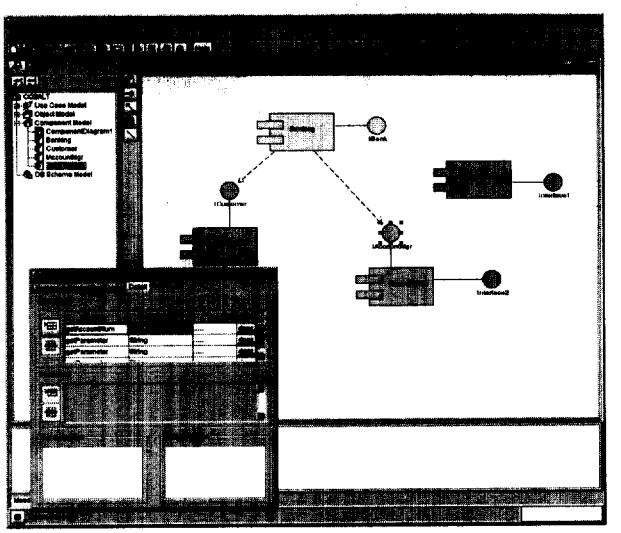
### 3.3 컴포넌트 다이어그램 작성[11]

우선 개념적인 컴포넌트 모델링 단계에서는 재사용을 고려하여 최대한 독립적으로 실행 가능한 모듈들로 설계하여야 한다. 컴포넌트들은 독립적으로 상세 설계 및 구현할 수 있어야 하므로 이 단계에서 컴포넌트간의 의존성을 명확히 기술하고 이러한 의존성을 컴포넌트의 인터페이스에 명확하게 정의하여야 한다. COBALT : Constructor 도구에서는 단일 인터페이스로 국한하지 않고 컴포넌트가 여러 역할을 가질 경우 여러 개의 인터페이스를 가질 수 있도록 허용하고 있다. 인터페이스를 좀더 명확히 식별하기 위해 컴포넌트간의 순차도를 작성하여 참조할 수도 있다. 컴포넌트의 인터페이스에는 컴포넌트에 접근하기 위해 사용되는 오퍼레이션들의 파라미터 및 리턴 타입 등의 명확히 기술되어야 한다. 오퍼레이션의 파라미터는 자바의 기본 데이터 형 뿐만 아니라 사용자 정의 클래스도 사용할 수 있다. 컴포넌트의 각 인터페이스는 상세 설계 시에 EJB의 인터페이스 세션 빈으로 변화된다. (그림 4)는 컴포넌트 다이어그램의 실행 예와 컴포넌트의 인터페이스 명세를 작성하는 과정을 보여주고 있다.

## 4. 컴포넌트 상세설계 [11]

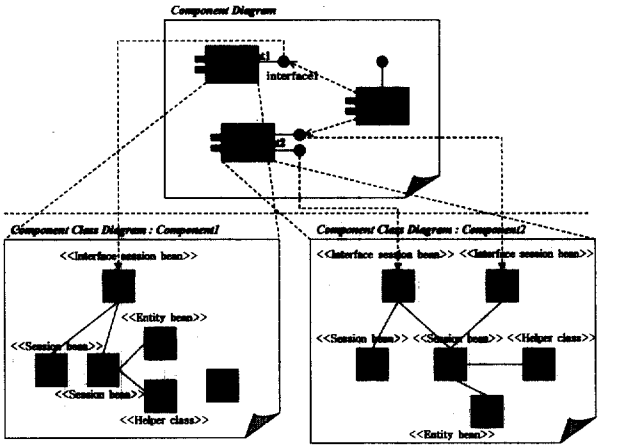
### 4.1 개념적 컴포넌트의 인터페이스 변환

개념적 컴포넌트 모델링 단계에서 정의된 각 인터페이스의 정보는 EJB 컴포넌트 상세 설계 단계에서 기능 연결을 담당하는 대표 세션 빈으로 (그림 5)에서와 같이 자동적으로 변환된다. 만약, 하나의 컴포넌트가 두개 이상의 인터페이스를 가질 경우 컴포넌트 클래스 다이어그램에 인터페이스의 개수와 동일한 인터페이스 세션 빈이 생성된다. 이 경우는 여러 역할에 따라 동일 기능을 하는 컴포넌트를 다시 설계하는 것보다는 내부 클래스를 공유하게 함으로써 보다 효율적으로 컴포넌트를 개발할 수 있다.



(그림 4) 컴포넌트 다이어그램 및 인터페이스 정의 화면

컴포넌트 상세설계 단계에서는 이러한 인터페이스 세션 빈을 바탕으로 컴포넌트 클래스 다이어그램을 사용하여 플랫폼에 의존적인 컴포넌트의 내부를 설계하게 된다. 상세설계 단계의 첫 번째 단계에서는 우선 컴포넌트의 인터페이스 정의 과정에서 생성된 인터페이스의 오퍼레이션들이 EJB 컴포넌트의 속성에 맞는 인터페이스 세션 빈으로 자동적으로 변경된다. 인터페이스 세션 빈은 외부에서 컴포넌트로의 접근을 담당하는 빈으로서 이 인터페이스 세션 빈을 통해 컴포넌트 내부에 접근할 수 있다.




(그림 5) 개념적 컴포넌트 모델링 및 EJB 컴포넌트로의 세분화

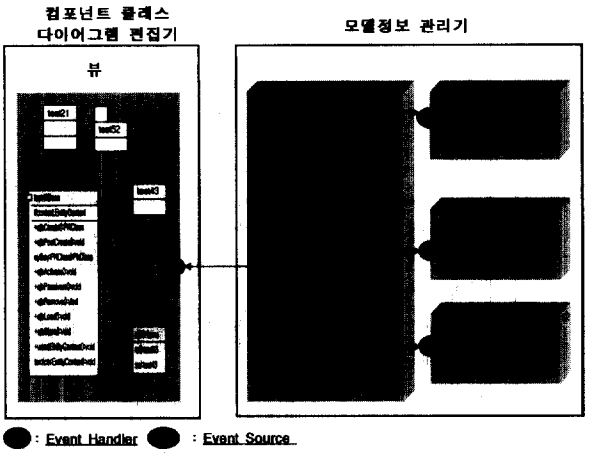
### 4.2 컴포넌트 클래스 다이어그램 편집기

컴포넌트 클래스 다이어그램 편집기는 각 컴포넌트를 구성하고 있는 클래스들의 정적인 구조를 표현하기 위한 편집기로서 컴포넌트 다이어그램에 기술된 컴포넌트의 요구사항에 맞게 클래스들을 정의하고 그들 간의 관계를 기술한다. 컴포넌트 다이어그램에 기술된 컴포넌트의 인터페이스를 컴포넌트 클래스 다이어그램에 표현하기 위해 대표 세션 빈(Session Bean)을 사용한다. 세션 빈의 내용이 변경되면 자동으로 컴포넌트의 인터페이스가 변경되고 반대의 경우도 마찬가지이다. 컴포넌트 클래스 다이어그램 편집기는 EJB 기반의 빈 클래스들을 정의하고 SRE를 지원하여 컴포넌트 클래스 다이어그램의 클래스와 파일 시스템의 자바 파일이 일대일 대응되도록 한다.

컴포넌트 클래스 다이어그램을 이용한 컴포넌트 상세설계에서 사용되는 표기법들은 기존 클래스 다이어그램에서 사용되는 표기법들을 확장하여 사용하였다. 또한 EJB 특성에 맞는 몇 가지 클래스를 추가하였다. 추가된 클래스는 외부에서의 컴포넌트로의 접근을 담당하는 인터페이스 세션 빈과 EJB 세션 및 엔티티 빈, DB와 직접 연결되어 있는 DB 엔티티 빈, 그리고 일반 클래스 등이 있다.

컴포넌트 클래스 다이어그램 편집기가 일반적인 클래스 다이어그램 편집기와 다른 점은 SRE를 지원하기 위해 다이어그램의 클래스 모델과 실제 자바 파일의 클래스 내용

이 동일하게 유지된다는 것이다. 컴포넌트 클래스 다이어그램의 클래스 내용이 변경되면 자바 파일의 내용이 변경되고, 자바 파일이 변경되면 변경 내용이 클래스 모델에 반영되어야 한다. 이를 구현하기 위해 컴포넌트 클래스 다이어그램은 (그림 6)과 같이 3 계층으로 이루어진다. 뷰는 다이어그램 편집기에 표시되는 부분으로 뷰 모델과 클래스 모델을 화면에 보여주는 역할을 담당한다. 뷰 모델은 뷰 정보를 저장하고 클래스 모델과 뷰를 서로 연결하는 역할을 담당한다. 클래스 모델은 다이어그램과는 별도로 클래스의 이름, 필드, 메소드 등의 논리적 정보만을 저장한다.

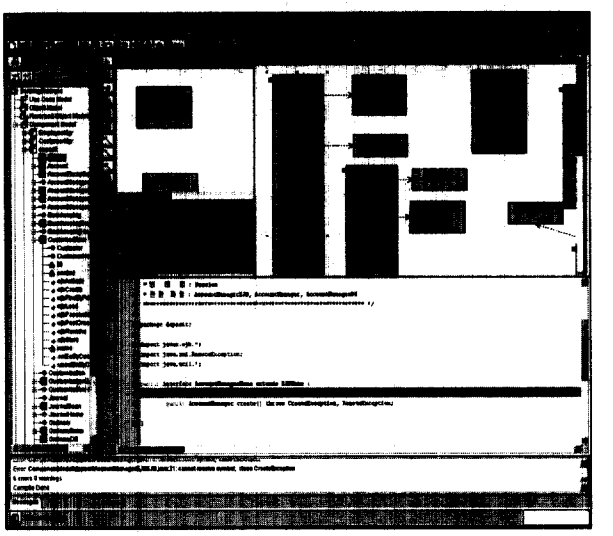



(그림 6) 컴포넌트 클래스 다이어그램의 구조

컴포넌트 클래스 다이어그램에서 하나의 노테이션을 화면에 보이는 과정은 다음과 같다. 첫째, 편집기에서 노테이션 생성 명령을 내린다. 둘째, 편집기에서 뷰 모델에 노테이션 생성 명령을 내린다. 셋째, 뷰 모델에서 클래스 모델에 클래스 생성 명령을 내린다. 넷째, 클래스 모델 생성 후 이벤트를 발생한다. 다섯째, 뷰 모델에서 클래스 생성 이벤트를 받아 뷰 모델을 생성 후 이벤트를 발생한다. 여섯째, 에디터에서 뷰 모델 생성이벤트를 받아 뷰를 생성한다.

이처럼 컴포넌트 클래스 다이어그램에서는 하나의 노테이션을 생성하기 위해 복잡한 과정을 거치지만 SRE 지원기와 같이 컴포넌트 클래스 다이어그램과 논리적으로 연결되는 다른 블록들은 클래스 모델에만 연결하면 되므로 블록간의 통합이 간단한 장점이 있다. SRE 지원기에 의해 클래스 모델이 생성되면 클래스 모델 생성 이벤트에 의해 뷰 모델과 뷰가 이벤트에 반응하여 새로운 클래스 노테이션을 화면에 표현한다.

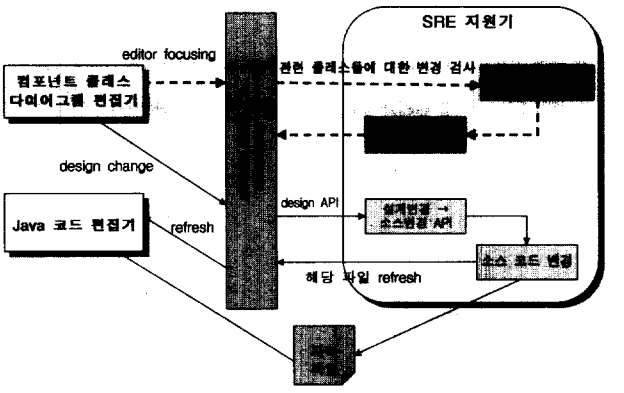
(그림 7)은 COBALT : Constructor 도구에서 컴포넌트 클래스 다이어그램 편집기를 이용하여 개념적 컴포넌트를 상세 설계하는 예를 보여준다. 왼쪽 아래의 다이어그램은 DB 모델러로서 클래스 다이어그램 형식을 이용하여 DB 스키마 모델을 표현하고 있다. (그림 7)에서는 DB 스키마 모델에 있는 Customer 테이블을 컴포넌트 클래스 다이어그램으로 가져오면 엔티티 빈인 CustomerBean으로 자동으로 변환한 예를 보여준다.



(그림 7) COBALT : Constructor에서의 컴포넌트 상세 설계 과정

### 4.3 SRE 지원 기능

SRE 지원기는 컴포넌트 클래스 다이어그램과 EJB 소스 코드의 일관성을 유지시키는 기능을 제공한다. SRE 기능은 크게 소스 코드 변경을 컴포넌트 클래스 다이어그램에 반영하는 기능과 클래스 모델 변경을 소스 코드에 반영하는 기능을 갖는다. 만약 소스 코드의 변경이 있었다면 해당 소스 코드를 분석하여 변경된 클래스 정보를 얻은 후에 모델 정보 관리기의 API를 이용하여 해당 클래스 정보를 변경한다. 컴포넌트 클래스 다이어그램이 변경된 경우에는 변경된 클래스 모델에 해당하는 소스 코드를 분석하여 클래스 모델의 내용에 따라 소스 코드를 변경한다. (그림 8)은 SRE 지원기의 구조를 보여준다.

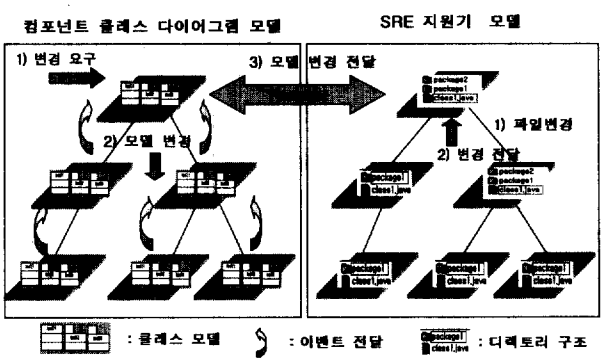



(그림 8) SRE 지원기의 구조

컴포넌트 클래스 다이어그램에서 클래스 모델들은 패키지 모델 내에 생성된다. 패키지 모델 역시 클래스 모델의 일종이므로 패키지 모델들은 트리 구조를 갖게 된다. 즉, 컴포넌트 클래스 다이어그램의 클래스 모델은 (그림 9)의

왼쪽에 나타난 것과 같이 패키지에 의한 트리 구조를 갖게 된다. SRE 지원기는 파일 시스템의 파일들을 분석하여 클래스 정보를 추출한다. 추출된 클래스 정보는 파일 시스템의 구조에 따라 (그림 9)와 같이 클래스 모델과 유사한 트리 구조를 갖는다.

컴포넌트 클래스 다이어그램의 클래스 모델과 SRE 지원기의 클래스 모델은 각각 별도의 클래스 모델을 트리 구조로 관리한다. 하지만 두개의 모델은 서로 동일한 구조와 동일한 내용을 유지해야 된다. 컴포넌트 클래스 다이어그램은 뷰 또는 다른 블록들의 요구에 의해 계속해서 모델이 변경된다. SRE 지원기 역시 도구와는 별도로 사용자에 의해 소스 코드가 변경될 수 있으므로 계속해서 모델이 변경된다. (그림 9)는 각 모델에 변경요구가 있을 경우 자신의 모델과 타 블록의 모델을 변경하는 순서를 나타내고 있다. 컴포넌트 클래스 다이어그램은 ClassModelManager가 모든 클래스들을 관리한다. 클래스가 생성, 삭제, 변경되면 이벤트가 발생되고 이 이벤트는 트리 상의 상위 노드로 전달된다. 결국 모든 트리의 변경은 루트 노드인 ClassModelManager에 전달된다. ClassModelManager는 전달된 변경 내용에 따라 SRE 지원기에 변경요구를 한다. 이 때 변경요구는 두 블록 사이의 약속된 프로토콜에 의해 이루어진다. SRE 지원기의 모델 역시 유사한 방법으로 자신의 모델을 변경하고 타 블록의 모델 변경을 요구한다. SRE 지원기는 자신이 관리하는 디렉토리를 계속해서 검사한다. 이때 변경된 파일, 또는 디렉토리를 발견하면 SRE 지원기의 모델을 변경하고 정해진 프로토콜에 의해 ClassModelManager에 모델 변경을 요구한다.




(그림 9) 컴포넌트 클래스 다이어그램과 SRE 지원기 모델간의 일관성 유지

컴포넌트 다이어그램 모델과 SRE 지원기 사이의 프로토콜은 다음과 같다.

- 패키지 단위 : 패키지 추가, 삭제, 변경
- 클래스 단위 : 클래스 추가, 삭제, 변경
- 클래스 내부 단위 : 속성메소드 추가, 삭제, 변경
- 연관성 단위 : 라인 추가, 삭제, 변경

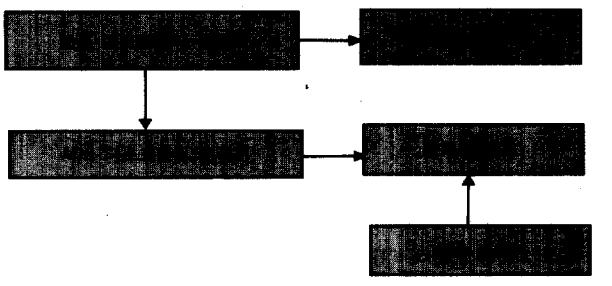
컴포넌트 클래스 다이어그램 모델과 SRE 지원기의 모델 사이에 프로토콜로 해결할 수 없는 경우도 있다. 예를 들어, 클래스 다이어그램이 존재하지 않는 특정 디렉토리를 역공학을 통해 클래스 다이어그램을 생성할 수 있다. 또는 옵션에 의해 SRE를 지원할 수도 있고 그렇지 않을 수도 있다. 이러한 경우는 트리 복사, 트리 비교 알고리즘을 적용하여 새롭게 모델을 생성한다. 컴포넌트 클래스 다이어그램과 SRE 지원기 모델간의 트리 복사 및 트리 비교 알고리즘에서는 트리의 각 노드에 특정 ID를 부여하여 두 모델의 노드간의 ID 값을 비교한다.

## 5. 컴포넌트 전개 및 테스팅 환경

### 5.1 컴포넌트 전개기

컴포넌트 전개기는 컴포넌트 생성 지원 도구를 이용하여 만들어진 컴포넌트 및 웹, 클라이언트 어플리케이션을 알맞은 DD(Deployment Descriptor)를 표현하는 XML 문서와 함께 패키징되어 J2EE를 지원하는 실제 어플리케이션 서버에 배치하는 역할을 담당한다. 일반적으로 컴포넌트 전개기는 J2EE를 지원하는 제품 고유의 도구로서 제공되며, 모델링 도구나 어플리케이션 서버를 지원하는 IDE는 일반적인 어플리케이션 서버에 대한 배치 기능을 가진다. 그러나, 어플리케이션 서버에 따라 J2EE 지원 정도와 고유 기능에 차이가 있어 하나의 모델링 제품 내에서도 전개 도구 역시 형태를 조금씩 달리하고 있다.

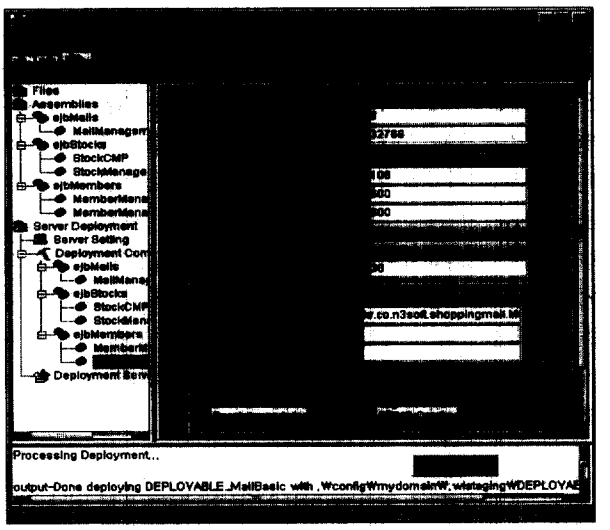
COBALT : Constructor의 전개기는 J2EE 1.2 Application (EAR), EJB 1.1, J2EE 1.2 Application Client를 기준으로 하여 작성되어 있으며 앞으로 EJB 2.0, Web Application 1.2(WAR), RAR도 지원할 예정이다. 제공되는 어플리케이션 서버는 현재 J2EE 1.2, J2EE 1.3, WebLogic 5.1, Web Logic 6.1로 추후 J2EE를 지원하는 대표적인 어플리케이션 서버를 지원할 계획이다.



(그림 10) 컴포넌트 전개기의 구조도

(그림 10)에서와 같이 사용자가 직접 컴포넌트 전개기를 실행할 때에는 독립형으로 실행이 되며 그 때에는 필요한 정보를 모두 입력 대화 상자를 통하여 입력받게 된다. 모델링 도구와 같은 타 도구에서 컴포넌트 전개기를 연결하고

자 하는 경우에는 컴포넌트의 기본 정보와 파일 리스트, 혹은 전개하고자 하는 표준 EJB 아카이브 파일을 넘겨주면 된다. 전개기는 기본 구성 정보(ejb-name, jndi-name 등)를 자동으로 구성하고 그 이외의 정보는 디스크립터 편집기를 통해 사용자가 직접 입력하게 된다. 전개기는 표준 아카이브 파일(jar, ear, war) 디스크립터를 편집하고 패키징하는 역할과 각 어플리케이션 서버에 전개할 수 있는 아카이브 파일 디스크립터를 편집하고 패키징 및 전개하는 역할을 해야 하므로 편집기도 두 가지 형태를 따로 가지고 있으며 패키징 관리기와 전개 관리기, 서버 관리기도 분리되어 있다. 이 중 전개 관리기와 서버 관리기, 서버 디스크립터 편집기는 어플리케이션 서버에 따라 별도로 구현하여야 하는 모듈이다.



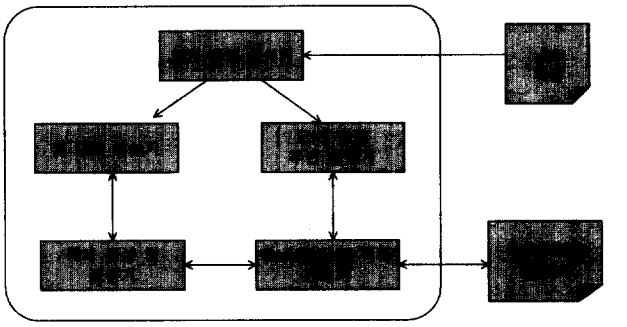

(그림 11) 컴포넌트 전개기의 실행 예

컴포넌트 전개기를 실행한 결과는 (그림 11)과 같다. File 메뉴의 주요한 기능은 New와 Option인데, New는 패키지를 처음 만들기 위한 메뉴이고 이 메뉴를 선택하면 패키지 종류부터 선택하는 대화 상자가 나온다. Option은 컴포넌트 전개기에서 사용되는 어플리케이션 서버를 선택하거나 항목 값을 바꾸는 데에 사용된다. 화면의 좌측 패널은 전개기 내의 모델 브라우저이다. Files 항목은 현재 선택된 패키지 종류와 전개될 때에 사용될 prefix와 디렉토리 명을 입력받게 되어 있다.

Assemblies 항목은 패키지 내의 컴포넌트들에 대한 디스크립터가 나열되어 있다. 각 디스크립터를 클릭하면 해당 디스크립터에 대한 편집기가 우측 패널에 나타난다. Server Deployment 항목은 어플리케이션 서버에 대한 내용이다. Server Setting은 현재 선택되어 있는 어플리케이션에 대한 값을 보거나 바꾸는 데에 사용된다. Deployment Configure의 각 항목을 클릭하면 우측 패널에 해당되는 서버 디스크립터 편집기가 나타난다. Deployment Server에서는 어플리케이션 서버에 전개된 컴포넌트 리스트를 보거나 전개 취소(undeploy) 시킬 수 있다.

### 5.2 컴포넌트 시험기

컴포넌트 시험기는 컴포넌트의 각 인터페이스의 기능을 검사하기 위해 컴포넌트 인터페이스의 오퍼레이션을 호출할 수 있는 웹 클라이언트 프로그램을 생성한다. 모델 정보 관리기로부터 JAR 파일을 입력 받아 JAR 파일을 분석하여 대상 컴포넌트의 정보들을 추출하고 이를 바탕으로 컴포넌트 인터페이스를 시험할 수 있는 JSP 클라이언트 프로그램을 자동 생성한다. 생성된 JSP 클라이언트는 사용자에 의한 컴파일 과정 없이 사용할 수 있다는 장점이 있다.

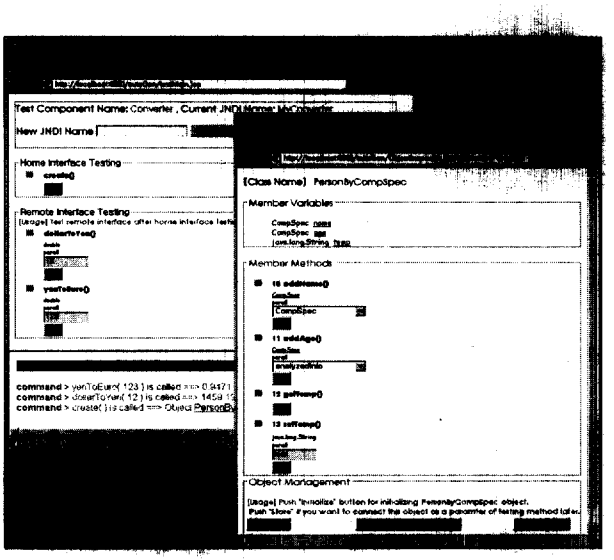


(그림 12) 컴포넌트 시험기의 구조

컴포넌트 시험기의 구조는 (그림 12)에 나타나 있다. 컴포넌트 시험기에서 웹 GUI 생성기는 (그림 13)에 나타나 있는 웹 기반의 클라이언트의 메인 GUI를 생성한다. EJB 연결 루틴 생성기는 실제적으로 어플리케이션 서버에 연결하여 EJB 컴포넌트를 수행하는 코드를 생성하는 부분이다. 그리고 사용자 정의 객체를 파라미터로 사용하는 메소드를 시험할 수 있도록 사용자 정의 클래스의 인스턴스를 생성할 수 있는 객체 생성 및 표현기와 생성된 객체 인스턴스들을 관리하는 Serializable 객체 관리기가 제공된다.

사용자는 어플리케이션 서버에 배치된 EJB 컴포넌트에 대해 "Component Test" 메뉴를 선택하면 컴포넌트 시험기에 의해 (그림 13)과 같은 컴포넌트 시험기 메인 화면이 자동으로 생성된다. 컴포넌트 시험 메인 화면에서는 Home 인터페이스와 Remote 인터페이스에 있는 각 오퍼레이션에 사용자가 직접 파라미터 값을 입력하여 각 오퍼레이션이 제대로 동작하는지 여부를 시험할 수 있다. 그리고 이 때 파라미터가 사용자 정의 객체인 경우는 (그림 13)의 오른쪽 창에 나타나 객체 표현기를 이용하여 새로운 객체 인스턴스를 생성하여 저장하고 다시 메인 화면으로 돌아와 저장된 객체를 읽어 해당 오퍼레이션을 수행할 수 있다. 객체 표현기는 객체 생성뿐만 아니라 리턴값으로 돌아온 사용자 정의 객체 인스턴스의 내용 확인에도 이용된다. 컴포넌트 시험기 메인 화면과 객체 표현기에는 각각 메시지 창이 있

어 사용자가 수행한 명령 및 결과 값을 보여준다.




(그림 13) 컴포넌트 시험기의 구조

## 6. 결 론

본 논문에서는 컴포넌트 생성 프로세스 전체를 지원하는 도구인 COBALT : Constructor 도구의 상세설계와 프로토타입의 구현에 대해 기술하였다. COBALT : Constructor 도구의 주요 특징은 전체 생성 프로세스를 체계적으로 연계하여 지원하는 점과 컴포넌트 모델링 단계를 플랫폼 독립적인 개념적 컴포넌트 모델링과 플랫폼 의존적인 컴포넌트 상세 설계 과정으로 나뉘어 지원한다는 점을 들 수 있다. 현재 COBALT : Constructor 도구의 프로토타입은 다이어그램 편집기들은 대부분 통합되어 있으며 SRE 지원기 등의 일부 블록은 현재 프로토타입을 개발 중이며 2002년 상반기에는 모든 블록이 통합되어 COBALT : Constructor 1.0 버전을 발표할 예정이다. 현재까지는 컴포넌트 상세 설계 부분에서 EJB 컴포넌트만을 고려하고 있지만 향후 버전에서는 DCOM, CCM 등을 반영할 계획이다.

## 참 고 문 헌

[1] Sun, Enterprise JavaBeans Specification, Version 1.1, Sun Microsystems Inc, 1999.

[2] OMG, CCM Revised Submission, OMG TC Document orbos/99-07-01, 1999.

[3] Microsoft, DCOM Technical Overview, URL : http : //msdn.microsoft.com/library/backgrnd/html/msdn_dcomtec.htm, 1996.

[4] Sun, Designing Enterprise Applications with the JavaTM 2 Platform, Enterprise Edition, Version 1.0, Mar. 2000.

[5] Aonix, "Component Factory," http : //www.aonix.com/content/products/select/compfact.html, 2001.

[6] Computer Associates, "COOL : Joe 2.0 Product Descriptions," http : //www.cai.com/products/cool/joe/cooljoe_pd.pdf, 2001.

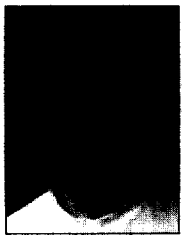
[7] TogetherSoft, "Features of Together," http : //www.togethersft.com/products/controlcenter/features.jsp, 2001.

[8] 윤석진, 신규상, "기존 프로그램에서의 독립 컴포넌트의 추출", 정보과학회 추계학술발표회, 제28권 제2호, pp.520-522, 2001.

[9] 조진희, 하수정, 김진삼, 박창순, "컴포넌트 기반 시스템 개발 방법론 마르미-Ⅲ," 프로젝트관리기술 논문집, 제4권 (통권 제4호), pp.1-13, 2001.

[10] 김민정, 이우진, 신규상, "개념적 컴포넌트 중심의 컴포넌트 모델링 기법 및 지원 도구의 설계", 정보처리학회 추계학술발표논문집, 제8권 제2호, pp.489-492, 2001.

[11] 정양재, 이우진, 신규상, "순환공학(Round-Trip Engineering)을 지원하는 클래스 다이어그램 설계", 정보처리학회 추계학술발표논문집, 제8권 제2호, pp.461-464, 2001.



### 이 우 진

e-mail : woojin@etri.re.kr

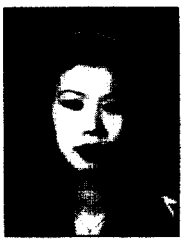
1992년 경북대학교 컴퓨터과학과 졸업(학사)

1994년 한국과학기술원 전산학과 졸업 (공학석사)

1999년 한국과학기술원 전산학과 졸업 (공학박사)

1999년~현재 ETRI 컴퓨터소프트웨어연구소 S/W공학연구부 선임연구원

관심분야 : CBD, Requirements Engineering, 웹서비스 기술, Petri nets, 실시간 시스템 모델링



### 김 민 정

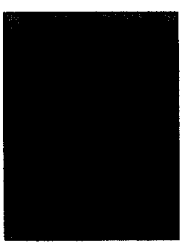
e-mail : minjkim@etri.re.kr

1998년 서강대학교 컴퓨터학과 졸업(학사)

2000년 서강대학교 컴퓨터학과 졸업(공학석사)

2000년~현재 ETRI 컴퓨터소프트웨어연구소 S/W공학연구부 연구원

관심분야 : CBD,에이전트 지향 소프트웨어 공학, 소프트웨어 아키텍처



### 정 양 재

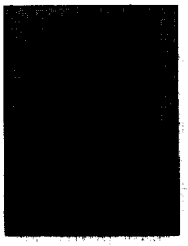
e-mail : comor@etri.re.kr

1999년 전북대학교 컴퓨터과학과 졸업 (학사)

2001년 전북대학교 대학원 전산통계학과 졸업(석사)

2001년~현재 ETRI 컴퓨터소프트웨어연구소 S/W공학연구부 연구원

관심분야 : 소프트웨어공학, 컴포넌트, 실시간시스템 등




## 윤 석 진

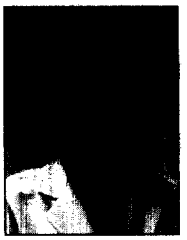
e-mail : sjyoon@etri.re.kr

1992년 중앙대학교 컴퓨터공학과 졸업 (학사)

1994년 중앙대학교 컴퓨터공학과 졸업 (공학석사)

1994년~현재 ETRI 컴퓨터소프트웨어 연구소 S/W공학연구부 선임 연구원

관심분야 : CASE, 컴포넌트, 재사용, HCI


## 이 지 현

e-mail : jihyun@etri.re.kr

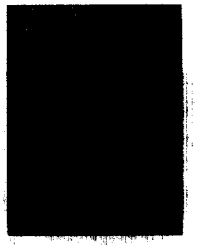
1998년 성균관대학교 정보공학과 졸업 (학사)

2000년 포항공과대학교 컴퓨터공학과 졸업 (공학석사)

2000년~2001년 LG전자기술원 정보기술 연구소 연구원

2001년~현재 ETRI 컴퓨터소프트웨어연구소 S/W공학연구부 연구원

관심분야 : 컴포넌트 기반 개발 방법론, S/W 재공학, 실시간 시스템 모델링


## 최 연 준

e-mail : june@etri.re.kr

1996년 부산대학교 전자계산학과 졸업 (학사)

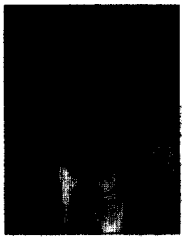
1998년 부산대학교 대학원 전자계산학과 졸업(이학석사)

1998년~1999년 데이텍㈜

1999년~2001년 ㈜아이티플러스

2001년~현재 ETRI 컴퓨터소프트웨어연구소 S/W공학연구부 연구원

관심분야 : 소프트웨어 공학, 웹서비스, J2EE 플랫폼 등


## 신 규 상

e-mail : gsshin@etri.re.kr

1981년 성균관대학교 통계학과 졸업(학사)

1983년 서울대학교 대학원 계산통계학과 졸업(이학 석사)

2001년 충남대학교 대학원 컴퓨터과학과 졸업(이학 박사)

1983년 시스템공학연구소 연구원

1987년 시스템공학연구소 선임연구원

1997년 한국전자통신연구원 책임연구원

현재 ETRI 컴퓨터소프트웨어연구소 S/W공학연구부 컴포넌트 공학연구팀 팀장

관심분야 : CASE, S/W 컴포넌트 기술, 웹 서비스 기술, 멀티미디어 시스템