□특별기고□

엔터프라이즈 환경에서 CBSD를 위한 비즈니스 컴포넌트

전북대학교 최성만* · 김정옥 · 유철중** · 장옥배**

1. 서 론

오늘날 빠르게 변화하고 있는 객체지향 기술을 대규모 시스템 개발에 적용하기 위해 출현한 컴포넌트 기술은 현재 소프트웨어 개발에서 가장 큰 이슈로 부상하였다. 정보화 사회의 발전속도를 만족시키기 위해 최근에 들어와 소프트웨어의 모습은 점점 더 복잡하고 대형화되어가고 있는 추세이다. 이러한 추세에서 소프트웨어 개발자나 공급자들이 사용자의 요구사항을 만족하는 서비스를 적재적소에 제공하기는 점점 어려워지고, 기존의 소프트웨어를 유지보수하는데에도 많은 어려움과 비용 및 시간을 필요로 한다. 이런 제약사항을 극복하기 위하여 새롭게 대두되고 있는 기술이 바로 컴포넌트 소프트웨어(component software)의 개발이다[1, 2]. 객체지향적 소프트웨어인 컴포넌트는 재사용성을 높여주며, 저비용 고효율의 장점을 가지고 있다. SI 업계에서는 이러한 컴포넌트가 급변하는 소프트웨어 개발 기술을 혁신적으로 변화시킬 것으로 기대하고 있다.

컴포넌트 소프트웨어 또는 컴포넌트웨어(componentware)를 가장 먼저 제안한 사람은 1985년 소프트웨어 IC를 주장한 Brad Cox이며, 이듬해인 1986년에는 StepStone사에서 ICpak101, ICpak201이라는 최초의 상용 컴포넌트가 개발되었다. 최초의 상용 컴포넌트는 Objective-C로 작성되었는데, 객체를 통한 소프트웨어의 재사용을 실현하였다[3]. 다양하게 정의되어진 컴포넌트 소프트웨어의 정의를 정리해보면 다음과 같다.

• Gartner Group

컴포넌트는 동적으로 바인드 할 수 있는 하나 이상의 프로그램들을 하나의 단위로 관리하는 패키지로서 실행시간에 발견될 수 있는 인터페이스를 통해 접근이 가능하다. 즉, 컴포넌트는 프로그램과 프로그램 사이의 메시지에서 파라미터를 정의하고 관리하는 것이 규칙적으로 구현되었기 때문에 개발자에게 친숙한 블랙박스이다[4, 5, 6].

• Component Software의 Clemens Szyperski

컴포넌트는 계약상으로 명시된 인터페이스들과 명확한 문맥 의존성을 가진 조합의 단위이다. 컴포넌트는 독립적으로 배치가 가능하고, 제 3의 업체의 컴포넌트에서도 이용할 수 있다[1, 4, 5].

• Rational Software의 Philippe Krutchen

컴포넌트는 잘 정의된 아키텍처 상에서 어떠한 기능을 수행하는 시스템의 독립적이면서 대치 가능한 부분이다. 컴포넌트는 인터페이스들의 집합에 대한 물리적인 구현을 제공한다[4, 5].

• SSA의 Kozaczynski

컴포넌트는 자발적인 비즈니스 객체 또는 비즈니스 로직을 소프트웨어로 구현한 것이다[4].

• Desmond Francis DSouza

컴포넌트는 소프트웨어 구현을 갖는 패키지라고 정의하면서 세 가지 특성을 제시하고 있다. 즉, 컴포넌트는 독립적으로 개발되고 보급될 수 있어야 하며, 컴포넌트는 인터페이스를 가져야 하는데, 자신이 외부에 서비스를 제공하는 제공 인터페이스(Provided Interface)와 자신이 필요로 하는 요구 인터페이스(Required Interface)를 가져야 한다. 또한 컴포넌트는 컴포넌트가 지니고 있는 속성을 커스터마이즈(cunstomizing) 할 수 있어야 한다[4, 7].

컴포넌트는 비즈니스 컴포넌트와 기능별 컴포넌트로 구분할 수 있는데, 본 고에서는 비즈니스 컴포넌트의 특징과 몇 가지의 중요한 개념을 분석하여

* 학생회원
** 종신회원

살펴본다. 비즈니스 컴포넌트가 어떻게 패키징(packaging) 되었는지를 나타내는 측면, 분산 컴포넌트가 비즈니스 컴포넌트로 구성되어져 사용되는 방법, 그리고 최종 사용자에게 기능을 제공하는 협력적인 방법과 비즈니스 컴포넌트의 독립성에 대하여 알아본다. 그리고 제시된 몇 가지 개념을 실제로 AddressBook 비즈니스 시나리오에 적용시켜보고, 비즈니스 컴포넌트 개념을 비즈니스 객체와 모듈 및 UML 패키지와 같은 다른 관련된 소프트웨어공학 개념과 비교해 본다.

본 고의 구성은 다음과 같다. 2장에서는 컴포넌트 입도의 레벨(levels of component granularity)에 대해서 살펴보고, 개념을 적용시킨 예제를 다루어본다. 3장에서는 비즈니스 컴포넌트의 내부 구조 즉, 비즈니스 컴포넌트의 네 계층의 개념과 특징에 대해서 알아보고, 각 분할 계층의 특징에 대해서 살펴본다. 4장에서는 비즈니스 컴포넌트의 외부 구조를 이루고 있는 구성요소인 플러그(plug)와 의존성(dependencies) 및 인터페이스(interfaces)에 대해서 언급한다. 5장에서는 전체 개발 라이프사이클에서 확장된 인터페이스의 개념을 살펴보고 개발 라이프사이클의 단계에서 비즈니스 컴포넌트가 표현하는 기법들에 대해서 알아본다. 6장에서는 주어진 각각의 개념들을 확장하여 AddressBook 비즈니스 시나리오에 적용하는데, 클라이언트/서버 시스템의 두 가지 관점으로 분석해 본다. 7장에서는 비즈니스 컴포넌트와 다른 관련된 소프트웨어공학 개념의 차이점을 비교해 보고, 마지막으로 8장에서는 결론 및 향후 연구과제에 대해서 논한다. 참고로 본 고에서 논하고 있는 내용은 주로 참고문헌 [8]의 내용을 참고로 하였음을 밝힌다.

2. 컴포넌트 입도의 레벨

비즈니스 컴포넌트의 접근법은 5차원(dimension)으로 구성되는데, 이것은 그림 1과 같이 랭귀지 클래스(language class), 분산 컴포넌트(distributed component), 비즈니스 컴포넌트(business component), 비즈니스 컴포넌트 시스템(business component system), 시스템 레벨 컴포넌트의 연동(federation of system-level components) 등으로 나눠진다.

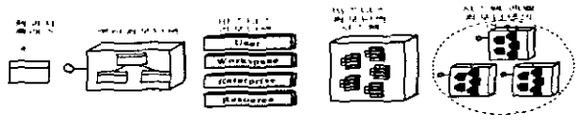


그림 1 컴포넌트 입도의 레벨

2.1 랭귀지 클래스

랭귀지 클래스(language class)는 일반적인 객체지향 프로그래밍 언어에서의 클래스를 의미하며, 매우 조밀한 속성을 가지는 랭귀지 클래스는 컴포넌트로 간주하지 않는다.

2.2 분산 컴포넌트

분산 컴포넌트(distributed component)는 컴포넌트에서 가장 낮은 입도 단계로서, 이것은 산업체에서 일반적인 컴포넌트 개념으로 EJB, COM, DCOM, CORBA 컴포넌트를 예로 들 수 있다. 특징으로는 잘 정의된 구축시간 및 실행시간 인터페이스를 가지며 실행시간 환경에서 독립적으로 플러그인 될 수 있다.

그림 2에서 보여주듯이 분산 컴포넌트는 다수의 랭귀지 클래스로 구성되는데 여기서의 분산 컴포넌트는 분산 컴포넌트 인터페이스를 통해 다른 프로그래밍 컴포넌트와 결합되어 구현될 수 있다.

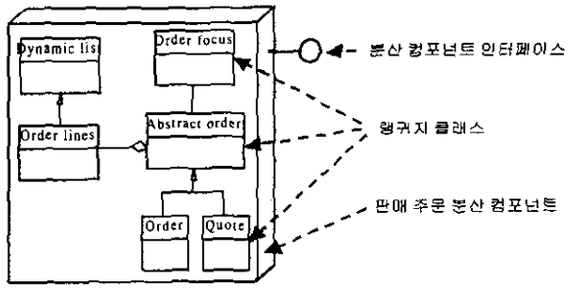


그림 2 랭귀지 클래스와 분산 컴포넌트

2.3 비즈니스 컴포넌트

비즈니스 컴포넌트(business component)는 자율적인 비즈니스 개념 또는 비즈니스 프로세스들의 소프트웨어 구현을 의미한다. 자율적으로 비즈니스 개념을 표현하며 구현 및 배치하는데 필요한 모든 소프트웨어 가공물들로 구성된다. 대개 하나 이상의 분산 컴포넌트로 구성되며, 둘 이상의 기계에 물

리적으로 배치될 수 있는 하나의 가공물이다. 비즈니스 컴포넌트는 사용자 계층(user tier), 워크스페이스 계층(workspace tier), 엔터프라이즈 계층(enterprise tier), 리소스 계층(resource tier) 등 네 개의 논리적인 계층을 가지는데, 그 특징은 모든 전달 가능한 개발 라이프사이클과 정보 시스템 구현에서의 비즈니스 개념이다. 각 개발 라이프사이클 단계에서는 비즈니스 컴포넌트를 특정 관점으로 다룬다. 결론적으로, 비즈니스 컴포넌트는 하나 이상의 분산 컴포넌트를 포함하는 소프트웨어 가공물의 집합으로 구성된다.

2.4 비즈니스 컴포넌트 시스템

비즈니스 컴포넌트 시스템(business component system)은 비즈니스 문제에 대한 해결책을 제시하기 위해 협력적으로 함께 조립된 비즈니스 컴포넌트들의 집합을 의미한다. 그 특징은 도메인 영역에서 특정 비즈니스 문제를 처리하며, 하나 이상의 명시가능하고 자율성을 가지는 비즈니스 프로세스로 구현된다. 전체 비즈니스 컴포넌트 시스템은 컴포넌트처럼 존재할 수 있는데, 그림 3은 벤더 송장 관리를 하는 비즈니스 컴포넌트 시스템의 예를 보여주며 각 박스는 개별적인 비즈니스 컴포넌트를 의미한다.

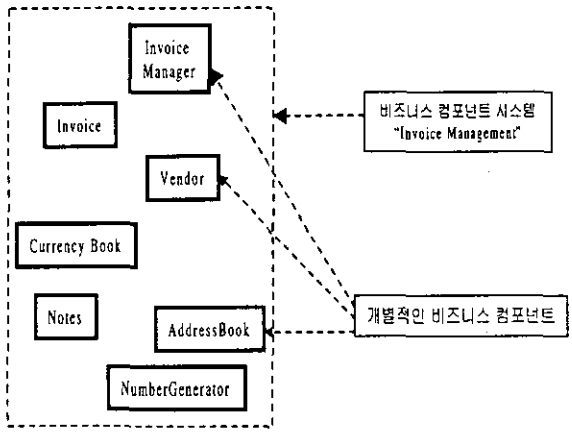


그림 3 비즈니스 컴포넌트 시스템의 예

2.5 시스템 레벨 컴포넌트의 연동

시스템 레벨 컴포넌트의 연동(federation of system-level components)은 서로 다른 조직에 속하는 최종 사용자들의 정보처리 요구를 다루기 위해서 협력하는 시스템 레벨 컴포넌트들의 집합을 의미한다. 그림 4에서는 게이트웨이(gateway) 컴포넌트가 보이지 않는 간단한 시스템 레벨 컴포넌트의 연동을 보여주고 있다. 만일 각 시스템 레벨 컴포넌트의 기법, 기반 구조, 비즈니스 컴포넌트, 애플리케이션 아키텍처가 동일하지 않으면, 각 시스템 레벨 컴포넌트 내부에 공통 비즈니스 컴포넌트가 존재하더라도 컴포넌트 15개 모두를 구현해야 한다. 그러나 동일한 비즈니스 컴포넌트가 있는 경우에는 10개의 비즈니스 컴포넌트만 구현하여 사용함으로써 개발비용을 감소시키는 효과를 얻을 수 있다.

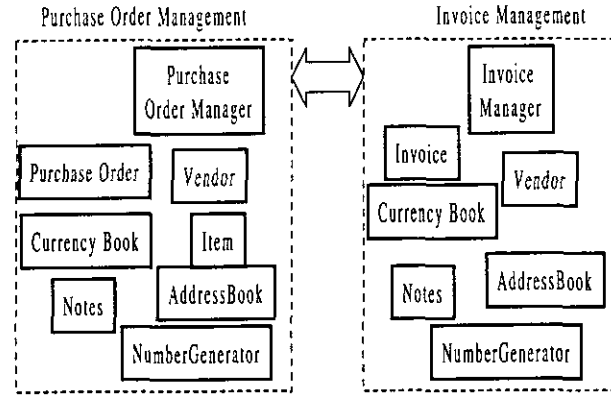


그림 4 시스템 레벨 컴포넌트 연동의 예

3. 비즈니스 컴포넌트의 내부 구조

비즈니스 컴포넌트의 상세한 분석(내부 아키텍처, 구조, 구성요소 부분)을 기능 설계자 및 개발자의 관점에서 알아보도록 한다.

3.1 분할 계층

기능 설계자 관점에서 단일 비즈니스 컴포넌트 하나는 그림 5와 같이 네 개의 연속된 계층으로 구

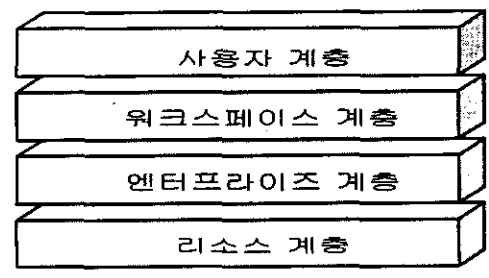


그림 5 비즈니스 컴포넌트의 네 계층

성된다. 각각의 계층들은 논리적인 책임 영역을 가지며 특별한 행위와 디자인 패턴 및 각각의 관심영역을 갖는다.

· 사용자 계층(User Tier : u-tier)

사용자 계층은 화면상에서 비즈니스 컴포넌트를 표현하며 사용자와의 대화에 대해서 모든 책임을 갖는 계층이다. 이 계층은 특별한 비즈니스 컴포넌트의 사용자 인터페이스 제어와 패널을 실행하고 구현하는 역할을 하는데, 모든 비즈니스 컴포넌트가 u-tier를 가지는 것은 아니다.

· 워크스페이스 계층(Workspace Tier : w-tier)

워크스페이스 계층은 사용자 계층을 지원하고, 오직 사용자를 지원하기 위한 비즈니스 로직으로서 엔터프라이즈 레벨의 리소스를 액세스할 필요가 없다. 웹 기반 아키텍처에서 w-tier는 인터넷 애플리케이션 서버에서 구현되며, 일반적인 비즈니스 컴포넌트의 w-tier는 다른 비즈니스 컴포넌트의 w-tier와 통신할 수 있다.

· 엔터프라이즈 계층(Enterprise Tier : e-tier)

엔터프라이즈 계층은 주어진 비즈니스 컴포넌트의 핵심 계층으로서 엔터프라이즈 레벨의 비즈니스 규칙, 확인(validation), 엔터프라이즈 컴포넌트간의 상호작용을 구현한다. 또한, 이 계층은 데이터의 무결성 측면을 관리하고 트랜잭션 서비스와 보안 및 지속성을 요구한다.

· 리소스 계층(Resource Tier : r-tier)

리소스 계층은 공유 리소스에 대한 물리적인 접근을 관리하는 계층으로서 실제적인 비즈니스 로직은 존재하지 않고, 공유 데이터베이스의 데이터를 가장 일반적인 리소스의 형태로 가진다. 이 계층에서는 다양한 데이터베이스 관리 시스템(Data Base Management System)의 특성에 관계없이 기능 개발자가 데이터베이스를 이용할 수 있도록 해준다.

3.2 분할 계층의 특징

비즈니스 컴포넌트의 역할과 관련된 특징들에 대해서 구체적으로 살펴본다. 그림 6에서는 비즈니스 컴포넌트 내에서 서로 다르게 상호작용 하는 분산 컴포넌트를 보여주고 있다.

DC에 의해서 발생된 이벤트에 대한 정보를 알려고 할 때, 점선에 의해서 보여준 것처럼 이벤트 발생을 인식하게 된다. 예를 들면, 그림 6에서 WDC의 요청에 따라 EDC가 이벤트를 발생시키는 과정을 표현하고 있다.

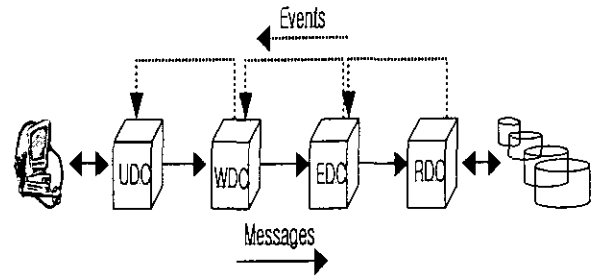


그림 6 내부-계층 상호작용

그림 7은 분산 컴포넌트들이 모여서 비즈니스 컴포넌트를 구성할 수 있는 여러 가지의 예를 유형별로 보여주고 있다. 비즈니스 컴포넌트는 보통 네 계층 모두로 구성되지만, 만약 하나 이상의 비즈니스 컴포넌트는 순차적으로 구성되어진다. 그림 7에서 보여주고 있는 처음 두 개의 컴포넌트는 가장 일반적인 형태의 비즈니스 컴포넌트로서, 첫 번째의 비즈니스 컴포넌트는 UDC와 WDC로 구성되고, 두 번째는 EDC와 RDC로 구성된다. 나머지 두 개의 비즈니스 컴포넌트는 WDC와 EDC로 구성되며, 또 다른 하나는 EDC로만 구성된 경우이다. 이러한 형태의 비즈니스 컴포넌트는 상황에 따라서 적절하게 이용하면 된다.

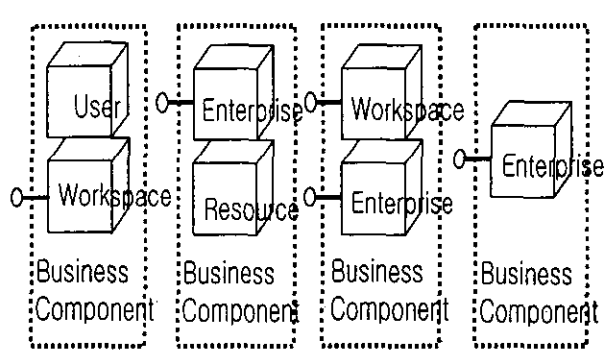


그림 7 비즈니스 컴포넌트의 예

4. 비즈니스 컴포넌트의 외부 구조

비즈니스 컴포넌트는 구축시간 및 실행시간의 개발 라이프사이클에서 각 단계별 외부 구조 분석이 필요한데, 그 이유는 독립적인 소프트웨어 컴포넌트를 생성해야 하기 때문이다. 비즈니스 컴포넌트의 외부 영역은 그림 8과 같이 인터페이스(interfaces), 종속성(dependencies), 플러그(plug)의 개념으로 구성된다.

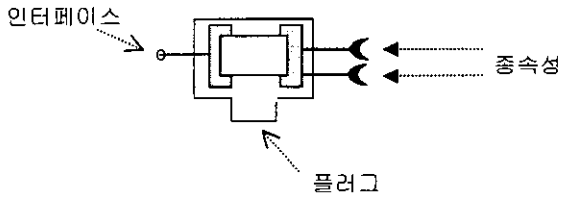


그림 8 비즈니스 컴포넌트의 외부 구조

4.1 인터페이스

비즈니스 컴포넌트의 인터페이스가 의미하는 것은 OMG의 IDL과 같은 인터페이스 정의 언어에 의해 명시된 실행시간 인터페이스에서 시작하고 개발 라이프사이클을 통해 그 의미는 점차적으로 확장된다[9]. 먼저 실행시간 인터페이스를 다루기 전에 그림 9에서 보여주듯이 DC 인터페이스의 가시성 레벨(levels of visibility) 차이점에 대해서 살펴본다.

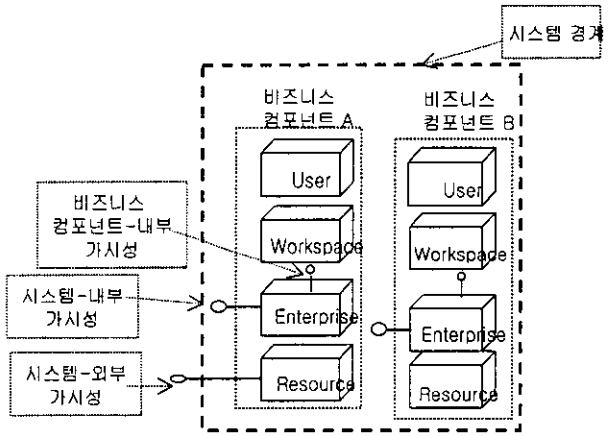


그림 9 비즈니스 컴포넌트의 가시성 레벨

· 비즈니스 컴포넌트-내부 가시성

이 인터페이스는 오직 비즈니스 컴포넌트의 내부에서만 볼 수가 있다. 즉, 그림 9에서 보여주듯이 비즈니스 컴포넌트 A의 EDC 인터페이스에서는 같은 비즈니스 컴포넌트인 비즈니스 컴포넌트 A의 WDC를 볼 수 있다.

· 시스템-내부 가시성

이 인터페이스는 시스템 경계 안의 비즈니스 컴포넌트 시스템에서 다른 비즈니스 컴포넌트를 볼 수 있다. 즉, 비즈니스 컴포넌트 A에서 비즈니스 컴포넌트 B를 볼 수 있다. 그러나 시스템 경계 밖의 비즈니스 컴포넌트는 볼 수 없다.

· 시스템-외부 가시성

이 인터페이스는 비즈니스 컴포넌트 시스템 경계 밖의 비즈니스 컴포넌트를 볼 수 있다. 이러한 인터페이스는 또한 시스템 게이트웨이 비즈니스 컴포넌트에 의해 제공되는 다른 특별한 외부 구조도 볼 수 있다.

지금까지 살펴보았던 인터페이스에 대한 가시성 레벨은 배치할 때 구성하여 할당할 수 있고, 개발하는 동안에 미리 정의되어질 수도 있다. 대부분의 컴포넌트 구현기술은 이러한 가시성 레벨을 직접적으로 지원하지는 않고, 프로젝트 규약(convention) 또는 도구를 통해서 지원하고 있다.

4.1.1 실행시간 인터페이스

실행시간 인터페이스(run-time interface)는 모든 시스템 내부와 외부 DC들간에 실행시간 인터페이스가 결합하여 이루어진다. 그림 10에서 보여주듯이 비즈니스 컴포넌트의 실행시간 인터페이스는 비즈니스 컴포넌트를 구성하는 EDCs와 WDCs간에 인터페이스 결합을 의미한다.

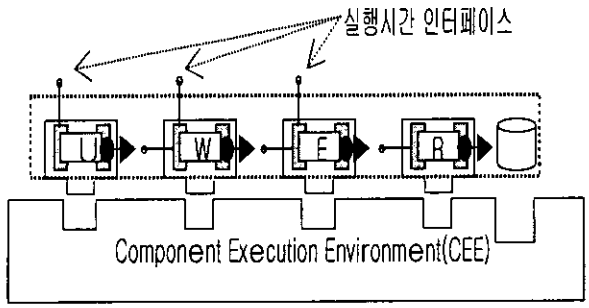


그림 10 비즈니스 컴포넌트의 실행시간 인터페이스

4.1.2 구축시간 인터페이스

구축시간 인터페이스(build-time interface)는

인터페이스 정의 언어를 통해서 설계시에 정의한 명세를 가지며, 실행시간 처리과정의 전통적인 의미를 넘어서 인터페이스 개념을 확장한 개념이다. 실행시간에 비즈니스 컴포넌트는 실행시간 기능 인터페이스를 통해 다른 컴포넌트를 볼 수 있는데, 그림 11과 같이 개발하는 동안에 비즈니스 컴포넌트 저장소에서 개발시간 인터페이스를 제공하므로 개발시에 기능 인터페이스를 볼 수 있다.

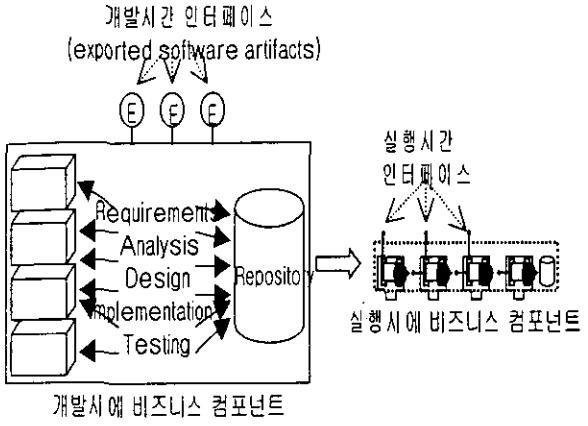


그림 11 비즈니스 컴포넌트 저장소

4.1.3 설계시간 인터페이스

설계시간 인터페이스(design-time interface)는 비즈니스 컴포넌트에 의해서 소유되거나 포함된 클래스 집합에 대한 명세서를 제공하거나 참조할 수 있도록 다른 컴포넌트의 컨텍스트에서 이용되어진다(그림 12). 여기서 클래스 집합은 타겟 DC의 프록시를 접근하는데 활용된다.

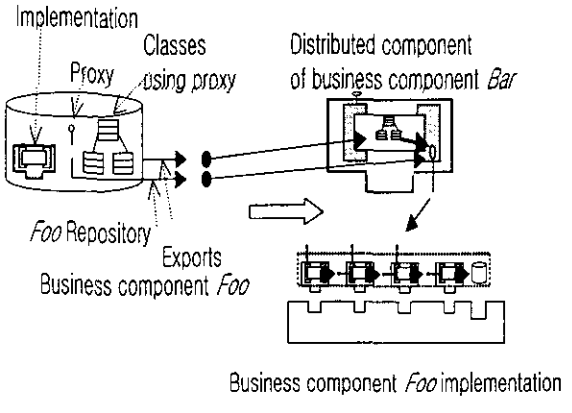


그림 12 설계시간 인터페이스의 예

4.1.4 개발 라이프사이클 인터페이스

컴포넌트의 확장된 인터페이스는 컴포넌트 접근점의 집합으로 표현될 수 있다. 컴포넌트의 접근점은 모든 분산계층과 구조적인 관점 및 컴포넌트 개발 라이프사이클의 모든 단계에서 컴포넌트가 개발될 때 산출된 소프트웨어 가공물 전체를 의미한다. 오늘날 확장된 인터페이스의 개념은 도구에 의해서 직접적으로 지원되는 경우가 많지는 않다.

4.2 종속성

그림 12에서 비즈니스 컴포넌트 Foo는 비즈니스 컴포넌트 Bar의 분산 컴포넌트로부터 소프트웨어 가공물을 요구할 때, 비즈니스 컴포넌트 Bar의 분산 컴포넌트에 소프트웨어 종속성을 갖는데, 이러한 종속성은 분산 계층과 개발 라이프사이클에서 발생될 수 있다. 컴포넌트 종속성에서 모든 소프트웨어 가공물을 찾기 위해서는 적절한 방법과 도구가 이용되는데, '*종속성 리스트(dependency list)*' 도구가 이용될 수 있다. 종속성 리스트는 비즈니스 컴포넌트가 의존하고 있는 소프트웨어 가공물과 DCs 그리고 모든 비즈니스 컴포넌트 집합들을 서술한다.

4.3 플러그

비즈니스 컴포넌트 외부 영역에서 플러그는 비즈니스 컴포넌트가 소켓에 의해서 제공하는 인터페이스를 만족시키기 위해 제공하는 소프트웨어 가공물의 결합이다. 저장소에서 플러그의 역할은 중요하며, 또한 BCVM(Business Component Virtual Machines)에서도 중요하게 작용한다. 예를 들면, 설계시간 비즈니스 컴포넌트를 새로운 환경에 플러그인 할 때 데이터베이스 스키마는 대형 시스템의 데이터베이스 스키마에 결합된다.

그림 13은 비즈니스 컴포넌트 소켓의 실행시간 일부를 보여주는데 각각의 계층은 적합한 소켓을 이용하여 플러그인 된다. u-tier에서는 사용자 인터페이스 프레임워크를 이용하여 플러그인 되며, w-tier에서는 Microsoft COM과 같은 로컬 CEE(Component Execution Environment)를 이용하여 플러그인 된다. e-tier에서는 EJB 모델 또는 CORBA 컴포넌트 모델과 같이 엔터프라이즈 레벨 컴포넌트 실행 환경에서 플러그인 되며, r-tier에서는 적합한 지속성 프레임워크를 이용하여 플러그인 된다.

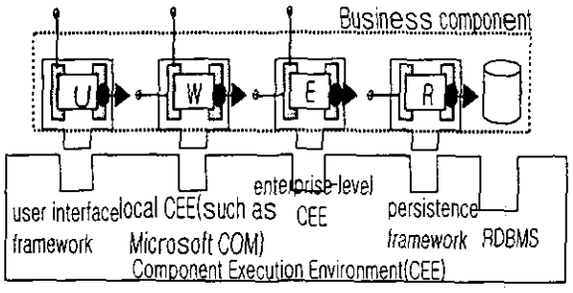


그림 13 비즈니스 컴포넌트의 소켓

5. 개발 라이프사이클

지금까지 비즈니스 컴포넌트의 외부 구조와 내부 구조에 대해서 분석해 보았다. 여기에서는 비즈니스 컴포넌트가 개발 라이프사이클의 여러 단계에서 표현되는 방법에 대해서 살펴보도록 한다. 모든 인도물들은 단일 비즈니스 컴포넌트의 단위로 서술되며, 각각의 개발 단계에서 산출되는 비즈니스 컴포넌트의 일부인 인도물들을 그림 14에서 보여주고 있다.

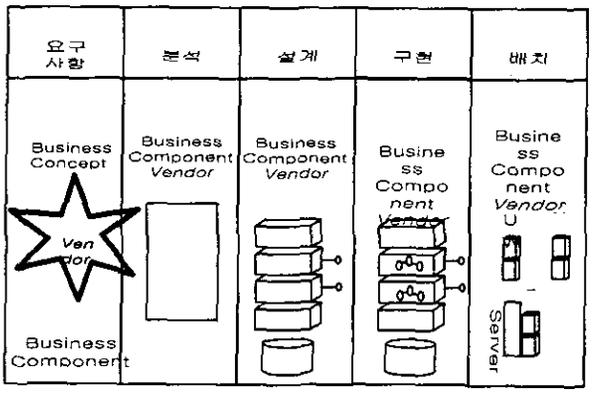


그림 14 개발 라이프사이클

· 요구사항

요구사항 단계에서는 비즈니스 컴포넌트를 설명하고, 비즈니스 컴포넌트를 다른 컴포넌트가 사용하는 방법을 예로 설명한 다이어그램을 요구하며, 기능 및 비 기능적 요구사항 모두를 포함해야 한다.

· 분석

분석 단계에서는 요구사항 단계에서 정의된 컨텍스트를 컴포넌트에 적용시켜 사용하는 컴포넌트 모델을 보여주고 필요에 따라서는 세분화한다.

· 설계

설계 단계에서는 모두 알려진 가공물의 상세한 정의 외에도 비즈니스 컴포넌트 자체의 세부 명세로서 구성된다. 즉, 다양한 분산 컴포넌트의 자세한 내부 구조 객체 모델을 포함해야 한다. 설계 단계에서 중요한 인도물은 비즈니스 컴포넌트의 종속성 리스트이다.

· 구현

구현 단계에서는 비즈니스 컴포넌트의 내부 구조에 대한 실질적인 코딩이 수행된다. 가장 중요한 분산 컴포넌트 구현 랭귀지 클래스와 분산 컴포넌트 인터페이스에 대한 프록시를 포함한다. 컴포넌트에 의해서 소유된 개념적이고 재사용 될 다른 컴포넌트의 다양한 소프트웨어 가공물을 포함하고 있다.

· 테스팅

비즈니스 컴포넌트 아키텍처는 상대적으로 분리된 상태에서 테스팅의 관점을 지원한다. 테스팅 단계에서는 일반적으로 테스트 계획, 테스트 사례, 테스트 데이터, 테스트 스크립트를 포함한다.

· 배치

배치 단계에서 비즈니스 컴포넌트는 여러 클라이언트의 서버 컴퓨터에서 실행할 수 있는 집합과 머신의 이용자 및 워크스페이스 계층을 표현하는 실행물의 집합으로서 표현된다.

6. AddressBook 시나리오

제시된 각각의 개념들을 확장하여 Address Book 비즈니스 시나리오에 적용하여, 클라이언트/서버 시스템의 두 가지 관점으로 분석한다. 첫째, 대형 시스템에서 단일 비즈니스 컴포넌트에 의해 제공된 사용자 인터페이스 이용하여 통합하고 둘째, EDC와 EDC간의 호출(다른 컴포넌트에 의해 호출되는 방법과 다른 컴포넌트에 의해 호출될 때 비즈니스 컴포넌트 내부에서 발생되는 내용)을 분석한다.

6.1 사용자 인터페이스

AdderessBook 비즈니스 컴포넌트는 속성 페이지를 제공한다. 그림 15와 같이 Vendor 비즈니스 컴포넌트가 사용자 인터페이스에 통합될 때, 속성 페이지는 다른 고려사항과 사용자 인터페이스 프레임워크, 설계, 구현기술 등의 결정이 실제 플러그

앤 플레이 관점에서 다소 쉽게 통합될 수 있다. AddressBook 비즈니스 컴포넌트는 사용자 인터페이스 프레임워크와 호환 가능한 속성 페이지를 전달해야 한다.

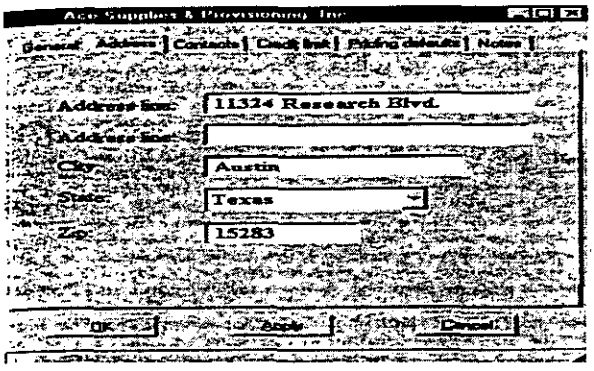


그림 15 벤더를 관리하는 속성 시트

그림 16의 순차 다이어그램은 e-tier에 의해서 제공된 갱신 데이터와 특정 사용자 인터페이스를 배치하는 과정을 보여주고 있다.

1 : 사용자가 벤더의 속성 시트의 주소 탭을 클릭한다.

2 : 주소 속성 페이지는 화면표시가 갱신될 필요가 있다는 것을 AddressBook u-tier에게 지시한다.

3 : AddressBook u-tier는 w-tier에게 벤더에 대한 현재의 주소 정보를 요구한다.

4 : 사용자 워크스페이스 도메인에서 EDC의 인터페이스 호출은 클라이언트측 데이터를 확인하지 않고 사용자에 의해서 입력된 데이터를 EDC가 원하는 형식으로 변환한다.

5 : e-tier는 데이터베이스에서 데이터를 로딩해서 확인할 필요가 있고, 대규모 시스템에서는 리소스 계층에 의해서 구현된 뚜렷한 분리 계층을 통해 실행된다.

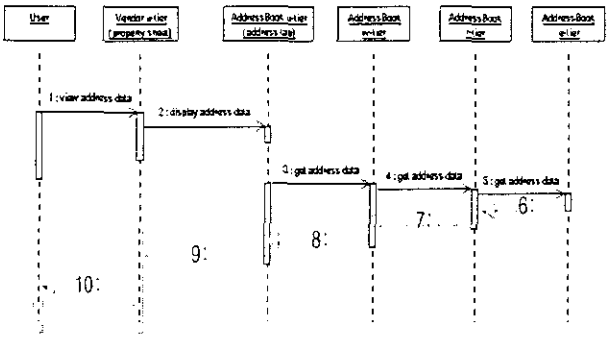


그림 16 사용자 인터페이스 예제에 대한 순차 다이어그램

6.2 EDC에서 EDC 호출

벤더에 대한 주소를 필요로 하는 시스템에서 송장을 출력하기 전에 송장에 대한 처리를 한다고 가정한다. 그림 17에서 보여주듯이 eInvoice EDC가 벤더에 대해 #123 송장을 출력하라는 청구서를 받았을 때, 벤더의 주소를 얻기 위한 eVender 컴포넌트의 호출은 EDC에서 EDC간의 호출 과정을 나타낸다.

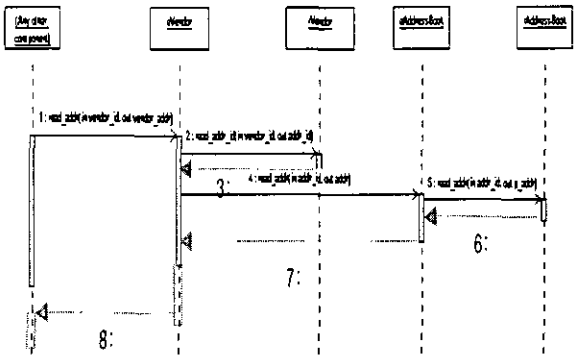


그림 17 EDC에서 EDC 호출

유일한 비즈니스 랭귀지 클래스, 즉 포커스 클래스를 가진 최소한의 주소록인 EDC의 내부 연산에 대해서 살펴보기로 한다. 그림 18은 eVendor에서 ieAddressBook(eAddressBook 분산 컴포넌트에 대한) 프록시를 가져오는 과정으로, 다이어그램의 뒷부분에서는 eAddressBook내에서의 intra-DC 상호작용을 보여준다. 즉, 인터페이스 구현 클래스

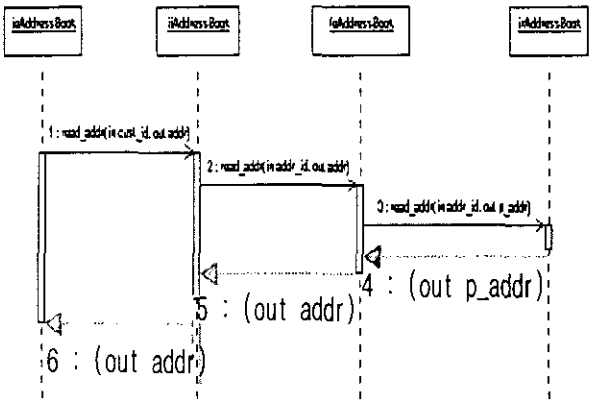


그림 18 Intra-DC 순차 다이어그램

(iieAddressBook), 포커스 클래스(feAddressBook), 리소스 계층에 대한 프록시(irAddressBook) 과정을 보여주고 있다.

7. 관련된 소프트웨어공학 개념

비즈니스 객체와 모듈 및 UML 패키지간의 차이점을 비즈니스 컴포넌트와 비교하여 분석해 본다.

7.1 비즈니스 객체와 모듈

여기에서는 비즈니스 컴포넌트와 비즈니스 객체, 모듈사이에서 개념상의 차이점을 분석하여 표 1로 표현하였다[2, 7, 8, 10].

표 1 비즈니스 컴포넌트와 비즈니스 객체, 모듈간의 개념 비교

구분	개념상의 차이점
비즈니스 컴포넌트	- 비즈니스 컴포넌트는 하나 이상의 분산 컴포넌트를 포함하는 소프트웨어 가공물의 집합
비즈니스 객체	- 두 가지 의미 · 비즈니스의 객체 · 구현 가공물을 의미
모 듈	- 두 가지 의미 · 논리적인 부분의 집합으로 문제 공간의 분할을 지시하는데 이용 · 관련이 없는 소프트웨어 가공물의 집합을 패키지화 하는 실행 파일 또는 코드의 덩어리

7.2 UML 패키지

UML에서 패키지는 요소를 그룹으로 구성하기 위한 일반적인 목적 메커니즘으로 정의한다. 구조적인 것 과 행위적인 것 및 그룹화 된 것까지 패키지로 배치될 것이다[2, 8, 11, 12]. 표 2는 패지지와 비즈니스 컴포넌트간의 유사점과 차이점을 나타내고 있다.

표 2 패키지와 비즈니스 컴포넌트간의 유사점과 차이점

관점 \ 구분		패키지	비즈니스 컴포넌트
유사점	메커니즘	클러스터링 메커니즘	클러스터링 메커니즘
	표현법	import의 형태로 표현	강력한 import/export 철학으로 표현
	내용	인터페이스와 구현물을 포함	접근법과 구현물 구조와 인터페이스의 종류를 포함
차이점	내포여부	내포됨	내포되지 않음
	내용	동일성과 시맨틱 범위를 포함하지 않음	동일성과 시맨틱 범위를 포함

8. 결 론

비즈니스 컴포넌트는 하나 이상의 분산 컴포넌트로 이루어진 컴포넌트이다. 본 고에서는 네트워킹이 가능한 인터페이스를 제공하기 위한 네 계층을 분석하였다. 개발자들은 각각의 역할과 기능들을 정확하게 요구 및 분석 명세로 정의하여 기업에서 원하는 최종의 시스템 구현에 용이하도록 적용하면 될 것이다. 비즈니스 컴포넌트를 구축함으로써 얻는 효과는 재사용성이 용이하고 비용을 절감할 수 있으며, 기존의 컴포넌트 접근을 용이하게 할 뿐만 아니라 인터넷 기반의 비즈니스 업무 처리 서비스를 폭넓게 지원할 것이다[13]. 컴포넌트 소프트웨어를 위한 개발여건은 객체 기술의 발전과 함께 충분히 진보했다고 볼 수 있다. 향후 연구과제로는 현재 소프트웨어 컴포넌트에 대한 표준이 없기 때문에 다양한 소프트웨어 컴포넌트를 하드웨어 플랫폼이나 운영체제에서 독립적이고 상호 호환성을 갖도록 하는 표준화에 대한 연구가 필요하며, 확장성과 플러그인을 위한 다양한 기술의 연구가 지속적으로 이루어져야 할 것이다.

참고문헌

[1] Clemens Szyperski, *Component Software : Beyond Object-Oriented Programming*, Addison Wesley Longman, Inc., 1998.

[2] Peter Herzum and Oliver Sims, *The Business Component Approach : Business Object Design and Implementation II : OOPSLA '96, OOPSLA '97, OOPSLA '98 Workshop Proceedings*, Eds. Eilip Patel, Jeff Sutherland and Joaquin Miller.London, UK:Springer-Verlag, 1998.

[3] Brad Cox. http://www.virtualschool.edu

[4] 김수동, 컴포넌트 정의 및 관련 기술 동향, 소프트웨어공학회지, 제12권 3호, pp. 3-18, 1999.

[5] 김상영, 황선명, 컴포넌트에 대한 IDL 기반 품질평가에 관한 연구, 정보과학회 춘계학술발표

논문집, 제27권 1호, pp. 546-548, 2000.
[6] John Cheesman and John Daniels, *UML Components - A Simple Process for Specifying Component-Based Software*, Addison Wesley Longman, Inc., 2000.
[7] Desmond F. D'Souza and Alan Cameron Wills, *Objects, Components and Frameworks with UML : The Catalysis Approach*, Addison Wesley Longman, Inc., 1999.
[8] Peter Herzum and Oliver Sims, *Business Component Factory : A Comprehensive Overview of Component-Based Development for the Enterprise*, John Wiley & Sons, Inc., 2000.
[9] Object Management Group (OMG), *OMG IDL Style Guide Document number: ab/98-06-03*. 1998b.
[10] Abadi, Martin and Luca Cardelli, *A Theory of Objects. New York*, NY : Springer-Verlag, 1996.
[11] Grady Booch, James Rumbaugh, Ivar Jacobson, *The Unified Modeling Language User Guide*, Addison Wesley Longman, Inc., 1999.
[12] Hans-Erik Eriksson and Magnus Penker, *UML Toolkit*, John Wiley & Sons, Inc., 1998.
[13] 김행곤, 최하정, 한은주, 분산 컴포넌트 명세에 기반한 비즈니스 컴포넌트 구축에 관한 연구, 정보과학회 추계 학술발표논문집, 제27권 2호, pp. 403-405, 2000.

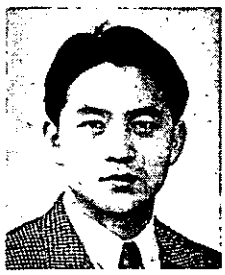
최 성 만

1999 전주대학교 전자계산학과 졸업 (이학사)
2000. 3~현재 전북대학교 대학원 전산통계학과 석사과정
1997~1999 정인대학 사무자동화과 조교
관심분야:소프트웨어공학, 객체 및 컴포넌트 기술, 에이전트 등
E-mail:sm3099@cs.chonbuk.ac.kr

김 정 옥

1985 전북대학교 전산통계학과 졸업 (이학사)
2000 전북대학교 정보과학대학원 컴퓨터정보전공 졸업(이학석사)
2000. 8~현재 전북대학교 대학원 전산통계학과 박사과정
2000 정보처리기술사
1985. 1~1990. 3 포항종합제철(주) 근무
1990. 4~1996. 5 한신생명보험(주) 전산실과장
1996. 5~ 1999. 3 전북도시가스(주) 전산실장
1997. 3~현재 정인대학 겸임 조교수
관심분야:소프트웨어공학, 객체지향 개발 방법론, 객체 및 컴포넌트 기술, 정보검색, 에이전트 등
E-mail:kjo3852@hanmail.net

유 철 중



1982 전북대학교 전산통계학과 졸업 (이학사)
1984 전남대학교 대학원 전산통계학과 졸업(이학석사)
1994 전북대학교 대학원 전자계산학과 졸업(이학박사)
1982~1985 전북대학교 전자계산소 조교
1985~1996 전주기전여자대학 전자계산과 부교수
1997~현재 전북대학교 컴퓨터과학과 조교수
관심분야:소프트웨어 품질 측정 및 보증, 객체 및 컴포넌트 기술, 에이전트 공학, 멀티미디어, HCI, 분산객체 컴퓨팅, 인지과학 등
E-mail:cjyoo@moak.chonbuk.ac.kr

장 옥 배

1966/1973 고려대학교 졸업(학사, 석사)
1988 산타바바라대 대학원(Ph. D.)
1974~1980 죠지아 주립대, 오하이오 주립대 박사과정 수료
1980~현재 전북대학교 자연과학대학 컴퓨터과학과 교수
관심분야:소프트웨어공학, 전산교육, 수치해석, 인공지능 등
E-mail:okjang@moak.chonbuk.ac.kr