특집

S/W 컴포넌트 조립도구 기반 조립 방법론 연구

정 연 대[1] 임 진 수[2] 오 연 재[2]

목 차

1. 서 론

현재 소프트웨어 개발 기술은 분산화, 통합화, 개방화, 부품화를 지향하고 있다. 이러한 S/W기술의 패러다임이 컴포넌트 기반 기술 환경을 요구하면서 J2EE, .NET, CORBA 등의 다양한 컴포넌트 플랫폼이 등장하고 있다. 소프트웨어 개발 기술도 급격히 변하고 있어 기술 변화에 쉽게 적용될 수 있어야 한다. 현재 Web Application Server에서 J2EE, .NET, CORBA기반의 S/W 컴포넌트는 Wrapping을 통하여 호환할 수 있는 수준으로 기술이 발전되어 있다. 각 플랫폼에서 컴포넌트를 생성하기만 하면 Web Application Server에서 COM+ 컴포넌트는 JCOM으로 Wrapping되어 EJB컴포넌트와 서로 연동이 가능하게 된다.

이러한 기술 환경의 발전으로 어떤 종류의 컴포넌트든지 일단 개발만 되면 컴포넌트 기반 시스템을 구축하는데 문제가 없는 것 처럼 보인다. 그러나 설사 컴포넌트가 잘 만들어져 있다 하더라도 개발된 컴포넌트를 plug-&-play 방식으로 조립하여 시스템 구축을 할 수 없으며, 제 3자에 의해 개발된 컴포넌트는 각 조직의 다른 비즈니스 프로세스 때문에 컴포넌트 인터페이스의 이름, 속성 및 메소드 등의 변경 없이는 조립을 통한 재사용이 불가능하다. 또한, 컴포넌트 기반 개발 시스템의 유지보수 문제도 어려운 실정이다. 현재 국내외에서 컴포넌트 개발이 활발히 진행되고 있으나 컴포넌트 유통이 활발히 진행되지 않은 것은 상기의 문제점에 기인한 이유가 크다. 이런 문제들을 해결하기 위해서는 아키텍쳐 기반의 컴포넌트 조립 프로세스가 확립되어야 하고 바이너리 컴포넌트 개조(Binary Component Adaptation) 기술[1, 2]이 개발되어야 하며 이러한 기능을 지원하는 컴포넌트 조립 지원 도구가 개발되어야 한다. 이렇게 되면 제 3자에 의해서 만들어진 컴포넌트라도 Plug-&-play 방식의 조립이 가능하고 소스코드의 수정없이 바이너리 컴포넌트의 수정이 가능하며, 시스템 아키텍쳐를 구성하는 컴포넌트간 연결 상태를 Visual하게 확인이 가능하므로 컴포넌트 기반 시스템의 유지보수시 컴포넌트 수정, 대체 및 삭제가 용이하게 된다.

1) (주)N3 SOFT 대표이사
2) (주)N3 SOFT

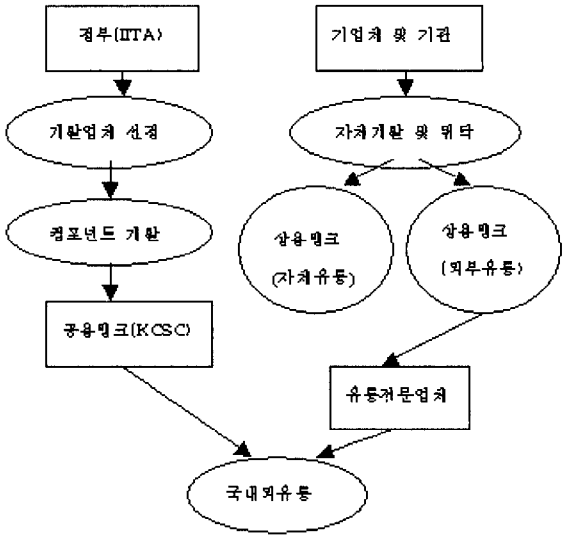
본 논문의 구성은 다음과 같다. 2장에서는 관련 연구로 국내외 S/W컴포넌트 개발 현황과 문제점들을 설명한다. 3장에서는 N3Soft가 개발한 컴포넌트 조립 방법론을 제안하고, 4장에서 EJB 컴포넌트 조립 도구를 설명하며, 5장에서는 제안한 컴포넌트 조립 방법론을 검증하기 위해 EJB컴포넌트 조립도구를 통한 간단한 쇼핑몰 개발 예제를 적용해 본다. 마지막으로 6장에서 결론을 맺는다.

2. 관련 연구

2.1 국내 S/W컴포넌트 개발 현황

2000년부터 국내 S/W컴포넌트 개발이 본격적으로 시작되어 현재 금융, 제조, 공공, 국방 분야 등으로 확산 되고 있는 추세다. 정보통신부가 ETRI와 함께 기획하여 만든 "컴포넌트 기반 기술개발" 과제를 중심으로 CBD(Component-Based Development) 방법론 개발[3]과 S/W컴포넌트 생성 및 조립 지원도구가 개발[4]되기 시작하면서 국내에 컴포넌트 산업이 도입 및 확산되게 되었다. 정부는 3년간 기업들에게 년간 30억원의 정부 재원을 투자하여 민간기업으로 하여금 보유하고 있는 S/W솔류션들을 컴포넌트화 하도록 유도하였고, 새로운 E-Business개발 기술들에 대해서도 자금을 지원하여 현재 30여개 업체가 300-400여개의 S/W컴포넌트들을 보유할 수 있는 수준에 이르고 있다. 이제는 정부의 지원없이 기업들마다 자발적으로 CBD를 도입하여 S/W컴포넌트를 만들어 나가고 있다.

국내 대표적인 SI기업인 삼성 SDS, LG C&S, 포스데이타 등은 자체 CBD방법론을 개발하여 자체 업무개발에 적용하고 있고, 중소 벤처기업인 Nexgen Tech, (주)엔쓰리소프트 등도 자체 CBD방법론인 ebCBD, N3_Methdology를 개발하여 용역개발과 컨설팅을 하고 있다. 금융분야에서는 수출입 은행이 가장 먼저 CBD를 도입하여 차세대 금융 시스템을 개발하였고, 뒤를 이어 조흥은행, 평화은행 등이 CBD를 도입 및 적용하여 S/W컴포넌트로 시스템을 개발하고 있다. 국민카드사가 EJB컴포넌트로 시스템을 구축한 후, 신용카드사들도 뒤이어 CBD를 도입하고 있다. 이동통신은 SK Telecom이 선두주자로 CBD를 도입하였고, KT, 보험사, 증권사들도 현재 S/W컴포넌트를 개발 중에 있다. 향후 주택은행과 통합한 국민은행이 대형 CBD프로젝트를 계획하고 있고, 국방분야도 재사용을 위한 시스템 구축의 하나로 CBD표준 방법론을 개발 중에 있어 주목을 받고 있다.



(그림 1) 상용 컴포넌트와 공용 컴포넌트의 유통 흐름

CBD방법론 기반의 S/W컴포넌트 개발 외에 그룹단위별로 컴포넌트 리포지토리(Component Repository) 시스템을 구축함으로써 자체 개발된 컴포넌트들을 저장, 검색, 관리 및 재사용을 극대화 하기 위한 기반 조성을 하고 있다. 결국 국내의 컴포넌트의 구성은 (그림 1)과 같이 정부 지원에 의해 개발되어지는 공용 컴포넌트와 업체 자체의 필요성에 의해 개발되어지는 상용 컴포넌트로 나

누어 볼 수 있다. 품질면에서 보면 공용 컴포넌트보다 바로 사용할 수 있는 상용 컴포넌트가 나은 편이다. 컴포넌트 재사용을 위한 정량적인 컴포넌트 품질 평가 기준이나 도구들이 없기 때문에 앞으로 품질보증 면에서 어려운 점이 많으리라 본다. 이제는 컴포넌트 재사용을 극대화시키고, 만들어진 수많은 컴포넌트들을 쉽고 빠르게 조립하고 유지보수 할 수 있는 방안들에 대해 연구를 해야 하리라 본다. (그림 1)에서 보는 바와 같이, 현재 국내에서 만들어지는 컴포넌트의 유통은 두개의 흐름을 가진다.

2.2 문제점 및 대안

S/W개발의 파라다임이 CBD로 옮아오면서 많은 분야에서 S/W컴포넌트를 개발해오고 있다. 현재 국내에서 개발되고 있는 CBD기반 과제의 문제점을 몇 가지 지적하면 다음과 같다.

첫째로, S/W컴포넌트는 재사용을 위한 조립 개념이 필수다. 그러나 현재의 국내 컴포넌트 개발 형태는 기존의 S/W개발과 별반 차이가 없이 개발이 이루어지고 있어 재사용 차원에서 어려움이 있다. CBD방법론을 보유하거나 컨설팅을 받는 회사를 제외하고는 대부분 EJB스펙이나 COM+ 스펙에 의존한 구현 위주의 개발이 대부분을 차지하고 있는데 이것은 국가적인 차원의 개발비 낭비를 초래한다. S/W 컴포넌트 개발의 본질을 잘 파악하여, 재사용 컴포넌트를 쉽고 빠르게 조립하는 방안을 연구해야 해야 한다.

둘째로, 컴포넌트를 재사용 하거나 조립하는 과정중에 컴포넌트의 개조가 필요하게 되는데, 이는 컴포넌트의 인터페이스가 조립하고자 하는 다른 컴포넌트와 다른 경우가 많기 때문이다. 특히, 제3자가 개발하여 유통되는 컴포넌트는 바이너리 형태로 소스코드가 없기 때문에 바이너리 컴포넌트 개조(Binary Component Adaptation, BCA) 기법[1, 2]의 컴포넌트 개조가 요구된다.

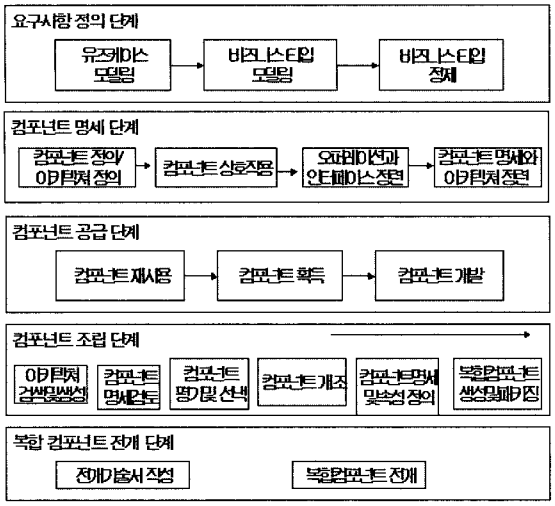
셋째로, (그림1)에서 처럼S/W컴포넌트는 캡슐화되어 유통되기 때문에 컴포넌트의 기능, 사용 방법, 사용환경, 사용 이력(Reuse History) 및 개발자 정보 등을 포함하는 컴포넌트 기술서가 함께 제공되어야 된다. 기술서에는 또한, CBD방법론에 의한 컴포넌트의 식별과 컴포넌트 시스템 아키텍쳐에 의한 관련 컴포넌트 간의 관계성 파악을 위한 정보도 포함되어야 한다.

넷째로, 컴포넌트 검색을 위한 리포지토리 시스템이 먼저 구축되어야 개발된 컴포넌트를 저장하고 관리하여 사용자가 필요한 컴포넌트를 즉시 활용할 수 있다. 이 시스템은 국가적 차원에서도 필요하지만 규모가 큰 대기업이나 기관들은 자체의 컴포넌트 리포지토리를 구축함으로써 자체 개발된 컴포넌트들을 사원들이 필요에 따라 저장과 검색을 쉽게 할 수 있어야 재사용을 극대화 시킬 수 있다. KT는 작년에 컴포넌트 리포지토리를 만들어 활용할 준비단계에 있고, 국방 분야도 리포지토리를 먼저 개발한 후, 각 군별로 개발되는 컴포넌트들을 체계적으로 종합하여 관리할 계획이다.

마지막으로 체계적인 재사용 가능한 컴포넌트를 인식하기 위한 CBD방법론의 개발 및 재사용을 위한 환경구축이 중요하다. 개발기간에 쫓기거나 예산의 부족으로 이러한 단계를 무시하게 되면 조립단계와 유통상의 어려움이 많아져 새로운 문제가 대두되고, 또다른 비용을 지불하게 되는 결과를 초래하게 된다. 본 논문에서는 4가지의 문제점 중 첫번째와 두번째의 문제점을 해결하기 위한 컴포넌트 조립방법론을 제안하고, 예제를 통해 기존 방법과 EJB 컴포넌트 조립도구에 의한 방법론의 장단점을 기술함으로써 조립도구의 필요성을 설명한다.

3. N3 컴포넌트 조립 방법론

본 장에서는 이미 개발된 컴포넌트들을 조립



(그림 2) N3 컴포넌트 조립 방법

하는 방법론을 제안한다. (그림 2)는 컴포넌트 조립 방법의 프로세스중 계획단계, 시스템 테스트 단계, 인도단계를 제외한 핵심적인 부분만 나타낸 것이다.

컴포넌트 조립 방법은 크게 다섯 단계의 작업으로 진행된다. 요구사항 정의 단계에서는 개발할 시스템의 영역을 이해하고, 시스템의 기능적, 비기능적 요구사항을 분석한 후 업무의 흐름과 비즈니스 타입을 도출, 정제한다. 컴포넌트 명세단계에서는 컴포넌트를 도출한 후 컴포넌트 시스템 아키텍쳐를 생성한다. 컴포넌트 시스템 아키텍쳐로부터 컴포넌트간의 상호 관계성을 검토하고, 컴포넌트부터 도출된 인터페이스와 오퍼레이션을 정제한다. 컴포넌트 공급 단계에서는 명세단계에서 정제된 컴포넌트 시스템 아키텍쳐로부터 컴포넌트 재사용 시나리오를 작성하고 필요한 컴포넌트를 획득하거나 필요한 컴포넌트가 없는 경우 새로 개발하는 대책을 세운다. 컴포넌트 조립 단계에서는 컴포넌트 공급단계에서 생성된 컴포넌트 시스템 아키텍쳐를 정제하고 각각의 컴포넌트 명세를 검토 후, 후보 컴포넌트(Alternative Components)를 평가하여 재사용할 컴포넌트를 최종 선택된다. 만약 최종 선택된 컴포넌트의 변경이 필요하면 컴포넌트 개조 작업을 한다. 변경이나 가변성 처리가 필요 없다면 컴포넌트 속성 및 명세를 정의하여 두개 이상의 컴포넌트와 연결을 시켜 복합 컴포넌트(Composite Components)를 생성한다. 복합 컴포넌트의 전개 단계에서는 조립된 컴포넌트의 전개 기술서(Deployment Descriptor) 작성과 실제 전개를 실시한 후, 클라이언트 프로그램으로 시스템을 실행시키게 된다.

3.1 요구사항 정의 단계

시스템을 개발하기 위해서 가장 먼저 수행해야 하는 일은 요구사항을 정의하고 시스템을 분석하는 것이다. 이러한 작업은 컴포넌트 생성단계와 비슷하다. 사용자의 요구사항을 분석한 자료를 이용하여 시스템의 영역을 정하고 비즈니스 관점에서 시스템을 정의한다. 비즈니스 객체를 식별하고 이들이 수행하는 기능들을 정의한 후, 객체들을 이용하여 수행하는 비즈니스 타입을 정의한다. 모델링된 결과는 정제 단계를 거쳐 조립에 사용될 컴포넌트를 식별하는데 이용된다. 각각에 대하여 살펴보면 다음과 같다.

3.1.1 유즈케이스 모델링

개발하고자 하는 시스템의 영역을 정하고 시스템의 기능적, 비기능적인 요구사항들을 반영하여 시스템을 분석한다. 컴포넌트 생성에서의 유즈케이스 모델링과 동일하다.

- 액터 추출: 시스템과 관련된 모든 액터들을 추출하고 적당한 액터명 기술
- 유즈케이스 도출: 업무를 트랜잭션 단위로 구분하여 유즈케이스를 도출한다.
- 유즈시나리오 작성: 유즈케이스별로 상세한 시나리오를 작성하고, 타 유즈케이스와의 관계성을 기술한다.

3.1.2 비즈니스 타입 모델링

비즈니스 타입 모델링에서는 유스케이스 분석 결과를 기반으로 객체들을 인식하고 유사 객체들을 그룹핑한 후 개념적인 클래스화를 시도하고 클래스들과의 관계성을 가지도록 다이어그램을 도출한다.

3.1.3 비즈니스 타입 정제

비즈니스 타입 정제는 비즈니스 타입 모델링에서 도출된 개념 클래스들에 대한 Operation들을 도출하고 클래스들과의 관계성을 정확히 확인하기 위한 객체간의 시퀀스 다이어그램을 작성해 봄으로서 개발하고자 하는 시스템을 올바르게 분석했는지를 알아내고 사용자의 요구사항을 잘 반영했는 지를 알아내기 위한 것이다.

3.2 컴포넌트 명세 단계

비즈니스 타입 모델링이 정제된 후, 주요 후보 컴포넌트를 도출하고 조립하고자 하는 컴포넌트를 이용하여 후보 컴포넌트를 정제한다.

3.2.1 컴포넌트 정의/ 아키텍쳐 정의

후보 컴포넌트의 도출은 비즈니스 타입 모델링의 결과를 이용하여 도출한다. 객체와 유스케이스와의 관계, 객체와 객체간의 관계를 통하여 후보 컴포넌트와 인터페이스를 결정하게 되는데, 이 때 4-계층 아키텍쳐에 기반하여 각 계층별 후보 컴포넌트를 도출하고 개념적인 컴포넌트 아키텍쳐 다이어그램으로 표시한다.

- Presentation계층: 사용자와 가장 밀접하게 연결되는 후보 컴포넌트들로 구성되는 계층이다.
- Business 프로세스 계층: 사용자 그룹에게 제공하는 서비스를 수행하는 후보 컴포넌트들로 구성되고 프레젠테이션 계층의 후보 컴포넌트와 미들웨어 계층의 후보 컴포넌트들과의 서비스를 주고 받는다.
- 미들웨어 계층: 모든 사용자에게 제공하는 서비스를 수행하는 컴포넌트들로 비즈니스 방법을 관리한다.
- 시스템 서비스 계층: EJB의 엔터티빈과 세션빈의 관리와 같은 물리적 저장소나 데이터베이스를 관리한다.

3.2.2 컴포넌트 상호 작용

3.2.1에서 만들어진 컴포넌트 아키텍쳐 다이어그램을 중심으로 후보 컴포넌트 정제 단계를 거친 후 컴포넌트 저장소에 저장되어 있는 컴포넌트를 검색하여 후보 컴포넌트와 유사한 명세를 가지는 컴포넌트를 찾아내고 후보 컴포넌트의 명세 정보를 컴포넌트 저장소의 검색 질의로 입력하여 원하는 컴포넌트를 식별한다. 조립을 위해 식별된 컴포넌트의 정보를 반영하여 후보 컴포넌트의 기본 인터페이스 정보와 단위를 정제한다. 이 후 최종 컴포넌트 아키텍쳐 다이어그램을 4계층 구조에 맞게 재작성한다.

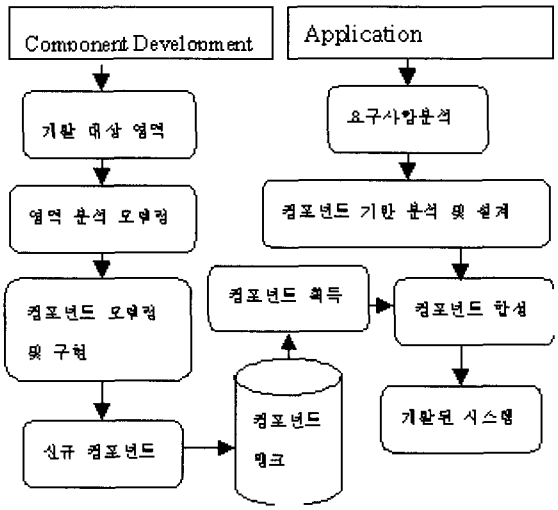
3.2.3 컴포넌트 명세와 아키텍쳐 정련

3.2.2에서 확정된 컴포넌트들에 대한 명세화 작업을 수행한다.

- 컴포넌트 개요
- 컴포넌트 내부 클래스도
- 컴포넌트 인터페이스 명세

3.3 컴포넌트 공급 단계

컴포넌트 시스템 아키텍쳐에서 나타난 필요한 컴포넌트들을 공급하기 위해서 기존 만들어진 컴포넌트를 컴포넌트 뱅크를 통해서 획득하여 재사용 여부를 결정하거나 필요한 컴포넌트가 없는 경우에 유통되는 컴포넌트를 구매하거나 새로 컴포넌트를 개발해서 공급을 해야 한다. 현재까지의 과정을 간략하게 그림으로 그려보면 (그림3)과 같다.



(그림 3)컴포넌트 개발 및 공급 흐름

3.3.1 컴포넌트 획득

컴포넌트 시스템 아키텍쳐를 참조하여 컴포넌트 뱅크로부터 필요한 컴포넌트를 검색하여 가져온다. 이때 컴포넌트 명세서를 참조하여 컴포넌트 식별단계에서 만들어진 명세서와 얼마나 일치하는지 확인하고 컴포넌트의 품질에 대한 평가를 한다.

3.3.2 컴포넌트 재사용

컴포넌트 획득과정을 통하여 획득된 컴포넌트를 바로 사용할 것인지, 부분적으로 수정하여 사용할 것인지, 새로 개발하여 사용하여야 할 것인지 판단하여야 한다. 재사용 단계에서 컴포넌트를 수정할 경우 바이너리 컴포넌트를 직접 수정할 수 없는 것이 일반적이나 당사에서 지원하는 컴포넌트 조립 지원 도구를 이용하여 컴포넌트 조립단계에서 바이너리 컴포넌트의 개조 작업을 통하여 속성, 인터페이스, Operation들을 수정할 수 있도록 목록을 작성한다.

3.3.3 컴포넌트 개발

컴포넌트 획득단계에서 필요한 컴포넌트가 없을 경우 컴포넌트 개발 프로세스를 통해 생성하도록 한다. 현재 컴포넌트 개발 방법론은 국내에서 개발된 마르미III[3]나 외국의 방법론[6, 7, 8, 9] 등 다양한 개발 방법론이 있으므로 이들을 참고할 수 있다.

3.4 컴포넌트 조립 단계

컴포넌트 조립 방법은 두가지로 구분하여 조립하는 방법이 있다. 첫째는 소스코드 수준에서 컴포넌트를 조립하는 방법과 컴포넌트 조립도구를 통해 조립하는 방법이 있다. 본 논문에서는 세계 최초로 개발된 아키텍쳐 기반의 컴포넌트 조립도구인 N3_Assembler를 사용하여 조립하는 단계를 설명하고 4장의 예제에서는 실제 조립되는 과정을 화면으로 제공한 후 두 방법의 차이를 분석하여 제공할 것이다.

3.4.1 아키텍쳐 검색 및 생성

컴포넌트 명세단계에서 생성된 컴포넌트 시스템 아키텍쳐를 검색하여 가져온 후 비지니스 컴포넌트를 중심으로 컴포넌트의 위치나 컴포넌트와 컴포넌트를 연결해주는 Connector를 지정하여 조립도구에 맞는 아키텍쳐를 재생성한다.

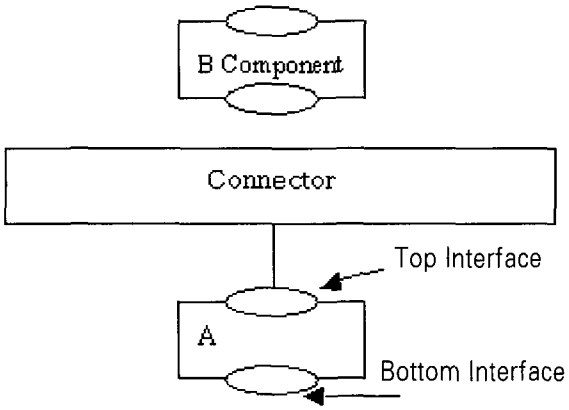
3.4.2 아키텍쳐 이해 및 정제

3.4.1의 과정을 통해 컴포넌트와의 관계성, Connector연결의 확실성에 대한 검토 및 이해를 하고 Top Interface나 Bottom Interface의 입력과 출력에 대한 자료를 참고하여 Bottom-Up형식의 아키텍쳐 모델을 구성하는 정제 작업을 수행한다.

3.4.3 컴포넌트 명세 검토 및 이해

아키텍쳐가 이해되고 정제된 후 각각의 컴포넌트간 정보(메세지)를 주고 받을 수 있도록 인터페이스의 명세를 정의해야 한다. 즉, 본 조립 방법론에서는 plug-&-paly 방식의 컴포넌트 합성을 구

현하기 위해 아키텍쳐를 구성하는 컴포넌트의 명세는 C2[5] 아키텍쳐 스타일을 기반으로 컴포넌트 인터페이스를 정의한다. (그림 4)는 C2 아키텍쳐 스타일의 구조를 보여주는 것으로 컴포넌트간 연결에서 보면 컴포넌트의 아래가 Bottom Interface이고 위가 Top Interface로서 하위에 있는 컴포넌트의 Top Interface에서 Connector를 통해 상위 컴포넌트의 Bottom Interface와 연결된다. 이런 연결을 위해서 각 컴포넌트가 가지고 있는 인터페이스의 명세들을 정의해 주어야 한다. 즉, Top 메시지 구성, Bottom메시지 구성, Behavior구성을 해준다.



(그림 4) C2 아키텍쳐 스타일의 구조

3.4.4 컴포넌트 평가 및 선택

컴포넌트의 평가 및 선택은 컴포넌트 명세 이해 단계에서 평가가 이루진다. 컴포넌트의 인터페이스 명세를 검토하는 단계에서 필요한 인터페이스 명세를 보유하고 있는지를 판단하고, 필요한 명세가 없으면 컴포넌트 개조 단계를 거쳐서 명세를 보완하도록 한다.

3.4.5 컴포넌트 개조

컴포넌트의 평가 단계에서 선택된 컴포넌트가 새로운 요구사항에 의해 필요한 명세를 확보하지 못하고 있을 경우 컴포넌트 개조기(Component Adaptor)를 통해 바이너리 컴포넌트를 개조하게 된다. 여기서 속성명 변경, 인터페이스명 변경, Method 변경, Method 추가 및 속성 추가를 통하여 재사용 컴포넌트의 요구사항 변경에 유연하게 대처할 수 있다.

3.4.6 복합 컴포넌트 생성

컴포넌트 시스템 아키텍쳐 정의에서 생성된 컴포넌트와 컴포넌트간의 연결이 완성되고 컴포넌트 명세의 이해와 개조가 모두 이루어지면 2개 이상의 컴포넌트가 결합된 복합 컴포넌트(Composite Component) 가 된다. 결국 여러 개의 컴포넌트가 연결되어 하나의 복합 컴포넌트로 구성되게 된다. 이 복합 컴포넌트는 저장 가능하며, 또한 다른 시스템 개발시 복합 컴포넌트가 사용되어지기를 원하면 저장소로부터 검색하여 하나의 단위 컴포넌트처럼 재사용될 수도 있다.

3.4.7 복합 컴포넌트 컴파일 및 패키징

3.4.6에서 복합 컴포넌트가 완성되면 하나의 컴포넌트처럼 활용하기 위해 컴파일 작업후, 패키징하면 새로운 하나의 jar파일이 생성되어 컴포넌트로서 전개가 가능한 상태가 된다.

3.5 복합 컴포넌트의 전개

대형 시스템의 경우 복합 컴포넌트들이 여러 개 만들어질 수 있다. 이런 경우 복합 컴포넌트를 하나의 컴포넌트로 인식하여 컴포넌트 조립을 할 수 있다. 여러 개의 복합 컴포넌트가 조립되어 하나의 시스템을 구축한 후, 컴포넌트 응용 서버(Component Application Server)에 전개하여 클라이언트와 연결하여 해당 시스템을 수행시킨다.

3.5.1 전개 기술서 작성

컴포넌트 조립의 전 단계가 끝나면 전체 시스템

구성에 맞게 전개한다. 복합 컴포넌트가 하나의 시스템을 이룰 경우 그 자체가 전개되는데 이때 각 컴포넌트의 개요, 인터페이스 명세, 배치도가 작성되어 시스템 전체의 조립 현황을 알 수 있도록 한다.

3.5.2 복합 컴포넌트 전개

조립의 최종 단계로서 전개기술서를 참조하여 실제 사용자가 복합 컴포넌트를 위저드 방식의 쉽고 편리한 전개기(Deployer)를 이용하여 컴포넌트 응용 서버(Component Application Server)에 전개 후, 클라이언트 프로그램을 사용하여 이를 실행한다.

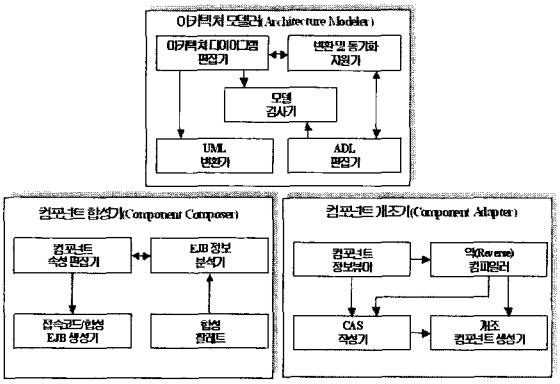
4. EJB 컴포넌트 조립 도구의 개요

4.1 EJB 컴포넌트 조립 도구의 구조

컴포넌트 조립 지원도구의 구조는 (그림 5)와 같다. 컴포넌트 조립 지원도구는 C2 아키텍쳐 스타일을 이용하여 시스템의 아키텍쳐를 설계하는 아키텍쳐 모델러와 이를 이용하여 생성된 아키텍쳐 모델을 바탕으로 새로운 합성 EJB를 생성하거나 독립된 응용프로그램을 생성해 주는 컴포넌트 합성기, 컴포넌트 개조의 원인별로 컴포넌트 개조 명세(Componet Adaptation Specification: CAS)를 작성하여 역컴파일(Decompile)된 원래의 컴포넌트 소스와 합쳐 새로운 개조 컴포넌트를 생성해주는 컴포넌트 개조기 등 3개의 서브 시스템으로 구성되어 있다.

4.2 EJB 컴포넌트 조립 도구의 기능

컴포넌트 조립 지원도구는 3개의 서브시스템과 13개의 세부 블록으로 구성되어 컴포넌트 조립 프로세스의 모든 과정을 지원하나 여기서는 주요 기능만을 설명하면 다음과 같다.



(그림 5)컴포넌트 조립 지원 도구의 시스템 구조도

4.2.1 아키텍쳐 모델러

4.2.1.1 아키텍쳐 다이어그램 편집기

아키텍쳐 다이어그램을 작성하거나 편집할 수 있도록 통합 GUI는 새 다이어그램을 생성하고 편집할 수 있도록 아키텍쳐 모델링에 관한 메뉴를 제공한다. 아키텍쳐 다이어그램 편집기에서는 컴포넌트와 커넥터의 편집 외에 합성 EJB 작성을 위해 필요한 속성편집, 명세편집, 인터페이스 편집 작업을 할 수 있도록 하여 컴포넌트 합성기와 연동한다.

4.2.1.2 ADL 변환기

그래픽의 아키텍쳐 다이어그램 모델 명세를 텍스트 형태의 ADL(Architecture Description Language) 명세로 변환하거나 역으로 ADL 명세를 아키텍쳐 다이어그램 모델 명세로 변환한다.

4.2.1.3 UML 변환기

아키텍쳐 다이어그램 편집기의 다이어그램 모델로부터 파싱된 ADL 명세를 입력으로 받아 이를 외부의 컴포넌트 생성지원도구에서 사용할 수 있는 형태로 변환한다. 구현이 없는 컴포넌트에 대해 외부의 생성도구를 이용하여 컴포넌트를 생성할 수 있도록 UML 표시를 따르는 컴포넌트 명세를 생성하여 이를 파일에 저장한다.

4.2.2 컴포넌트 합성기

4.2.2.1 합성 팔레트

아키텍쳐 모델 관련 파일, Java 화일, Jar 파일, 개조 관련 파일, XML 파일 등을 합성 팔레트 창에 보이고 파일 필터링(Filtering), EJB 컴포넌트 개조, 컴포넌트 시험, 전개, 선택 파일의 코드 편집의 기능을 제공한다.

4.2.2.2 EJB 분석기

EJB 정보 뷰어에서 EJB 분석 정보를 사용자에게 보이기 위해 EJB 패키지 파일 정보를 받아 DD(Deployment Description)파일을 분석하고 이를 바탕으로 홈/원격 인터페이스의 시그니쳐 정보를 얻는다.

4.2.2.3 속성 편집기

컴포넌트에 관한 속성 정보를 편집하는 기능을 하며, 명세 파일 편집을 위한 명세 편집기, EJB Wrapper를 편집하기 위한 코드 편집기, EJB 정보를 보기 위한 EJB 정보 뷰어와 연결된다.

4.2.2.4 명세 편집기

명세 편집기는 명세 정보를 보거나 편집하는 기능과 분석된 EJB 정보를 볼 수 있도록 하는 기능을 지원한다.

4.2.2.5 접속 코드 및 합성 EJB 생성기

아키텍쳐 다이어그램 편집기로 저장된 아키텍쳐 모델 정보 중 컴포넌트들의 EJB Wrapper 링크 정보와 아키텍쳐 형세(Topology) 정보를 바탕으로 사용자 지정 디렉토리(Directory)에 접속 코드(Glue Code)를 생성하고, 합성 EJB의 인터페이스를 편집하여 합성 EJB 빈, 홈/원격 인터페이스를 생성한다.

4.2.3 컴포넌트 개조기

4.2.3.1 컴포넌트 개조 명세(CAS) 작성기

특정 컴포넌트에 대한 어트리뷰트와 인터페이스에 대한 추가 정보 및 메소드/어트리뷰트/인터페이스의 이름 변경 정보를 입력 받아 개조 명세를 작성한다.

4.2.3.2 컴포넌트 정보 뷰어

CAS 작성기가 활성화되었을 때, 개조 대상 컴포넌트를 구성하는 파일 요소를 보여 준다.

4.2.3.3 역컴파일러

역컴파일러는 컴포넌트의 개조 시에 컴포넌트의 소스 코드가 없이 바이너리 클래스만 존재할 경우, 소스 코드를 생성하는 기능을 담당한다. 이는 EJB 컴포넌트의 경우 클래스 로더(Class Loader)의 수정이 불가능하기 때문에 실제 코드 생성 후, 이 코드에 컴포넌트 개조 명세의 내용을 반영해야 하기 때문에 필요하다.

4.2.3.4 개조 컴포넌트 생성기

개조 컴포넌트 생성기는 CAS 파일의 내용을 기존의 컴포넌트에 반영하여 새로운 코드를 만들고 이를 패키지화 한다. CAS 작성기로 만들어진 CAS 명세 파일을 CAS 분석기에 의해 분석하여 실제 개조된 정보를 식별하여 이를 역컴파일러가 만들어낸 기존의 코드에 반영하여 새로운 코드를 생성하고, 이를 전개기에 전달하여 EJB 패키지를 만들어 낸다.

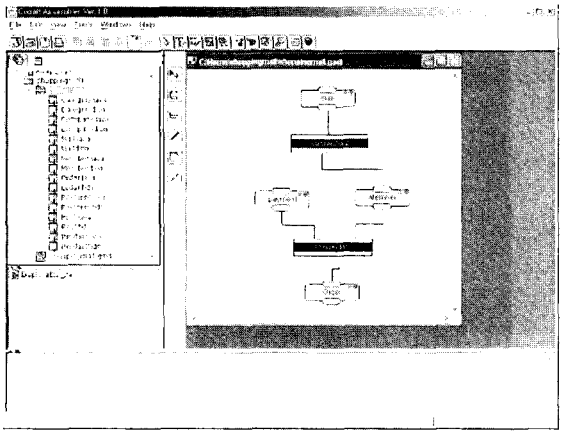
5. 컴포넌트 조립 방법론과 도구의 적용: 컴포넌트 조립 예제

본 장에서는 3,4장에서 기술된 방법론과 도구를 활용하되 방법론의 요구사항 정의단계, 컴포넌트 명세 단계, 컴포넌트 공급단계는 생략하고, 만들어진 컴포넌트를 재사용하기 위한 컴포넌트 조립

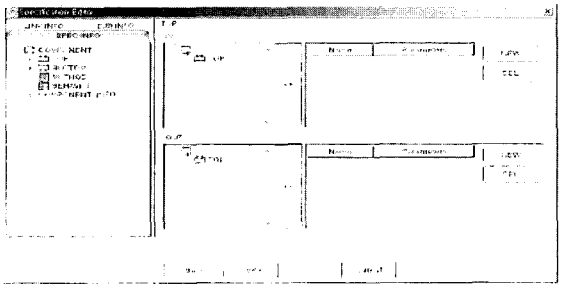
단계 부터 적용해보도록 한다. 컴포넌트 조립 방법을 간단한 쇼핑몰 예제에 적용함으로써 검증해 본다. 구매자가 쇼핑몰 사이트에 접속하여 상품을 조회하고 원하는 상품을 주문하는 시나리오를 컴포넌트 조립 방법에 적용하며, 조립에 사용되는 컴포넌트는 서버 레벨의 EJB 컴포넌트들이다.

5.1 아키텍쳐 검색 및 생성

위의 그림에서 왼쪽 창이 합성 팔레트를 구성하는 창으로서 재사용 컴포넌트의 모든 정보를 가지고 있다. 이 정보를 이용하여 오른쪽의 다이어그램 팔레트 창으로 정보를 옮길 수 있으며, 직접 아키텍쳐 다이어그램을 그릴 수 있다. 여기서 컴포넌트 아키텍쳐 다이어그램을 참조하여 컴포넌트 간의 관계성을 고려하여 최적의 아키텍쳐 모델링을 한다. 업무 전문가의 도움을 받거나 인터페이스 명세를 참조하여 다이어그램의 위치를 옮기거나 추가, 삭제도 가능하다. 아키텍쳐 모델링의 장점은 컴포넌트의 위치를 개발자들이 직접 눈으로 확인할 수 있고, 필요에 따라 필요한 컴포넌트를 검색하여 교체가 가능하도록 구성되어 있어 유지보수 및 컴포넌트 관리가 아주 쉽다.

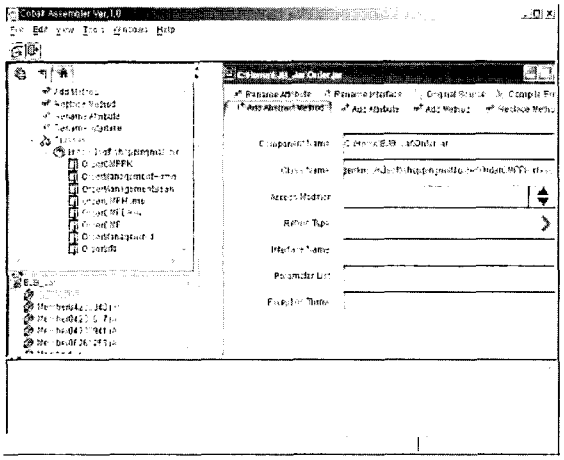


5.2 컴포넌트 명세 검토 및 이해



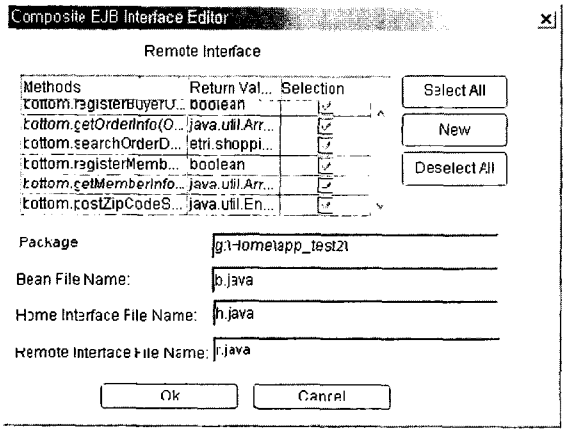
위의 그림은 명세 편집기로서 아키텍쳐 다이어그램에서 임의의 컴포넌트를 선택하여 그 컴포넌트가 가지고 있는 Top interface와 Bottom interface의 명세들을 검토 및 이해를 한 후 method들을 선택하거나 직접 입력하여 생성할 수 있다.

5.3 컴포넌트 개조



아키텍쳐 생성이나 컴포넌트 명세 검토단계에서 컴포넌트의 인터페이스나 속성, Method들을 수정하여 사용하기를 바란다면 컴포넌트 개조기에서 수정하여 컴포넌트를 다시 컴파일 한 후 재사용할 수 있다.

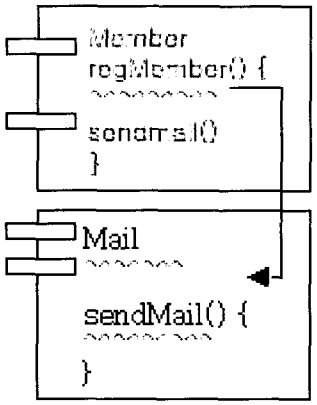
5.4 복합 컴포넌트 생성



5.5 기대효과

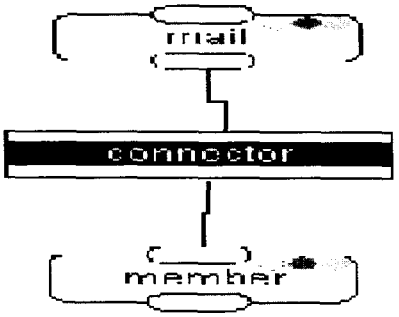
5.5.1기존 조립방식의 문제점 과 해결점-1

문제점 : 컴포넌트가 다른 컴포넌트의 인터페이스 를 직접 호출



다른 컴포넌트의 Interface를 호출하는 컴포넌트는 재사용하기 어렵다. 이 경우 Member 컴포넌트의 재사용이 어렵다. 또 조립관련 코드가 각 컴포넌트 마다 포함되어 있기 때문에 유지보수 할 때 연관된 모든 컴포넌트의 소스를 분석하고 필요할 경우 수정해야 한다. 이 경우 Mail컴포넌트 수정시 Mail과 연관된 Member 컴포넌트를 분석하여야 한다.

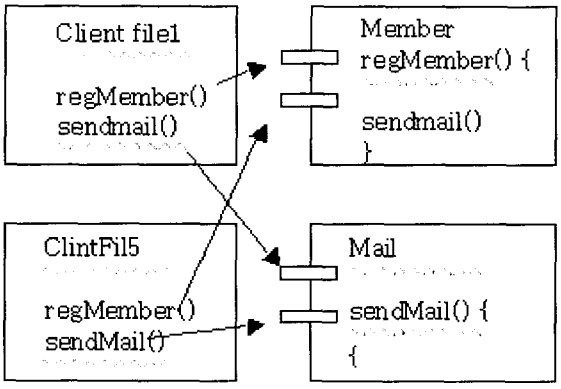
해결점 : 커넥터를 통해 컴포넌트간에 메시지를 주고 받음



컴포넌트가 다른 컴포넌트의 인터페이스를 직접 호출하지 않으므로 재사용이 가능하다. 컴포넌트가 수정되어도 메시지만 변경하면 되므로 유지보수가 수월하다. Mail컴포넌트 수정시 해당 메시지와 그와 연관된 메시지만 수정하면 된다.

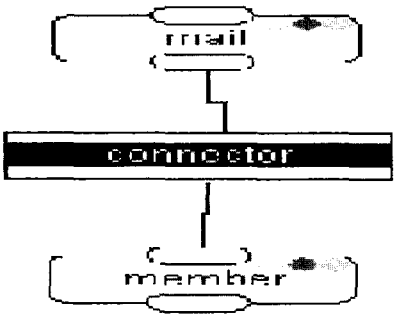
5.5.2 기존 조립방식의 문제점 과 해결점-2

문제점 : 컴포넌트간의 호출을 모두 없애고 클라이언트 에서 컴포넌트를 호출



컴포넌트간의 관계가 끊어져 컴포넌트의 재사용은 가능하지만 조립코드가 여러 클라이언트 파일에 들어가 있으므로 클라이언트가 복잡해 진다. 유지보수시 클라이언트 전반을 재검토해야하며 동일한 조립코드의 경우 모두 수정해 주어야 한다.

해결점 : 클라이언트는 조립된 컴포넌트의 인터페이 스만 호출

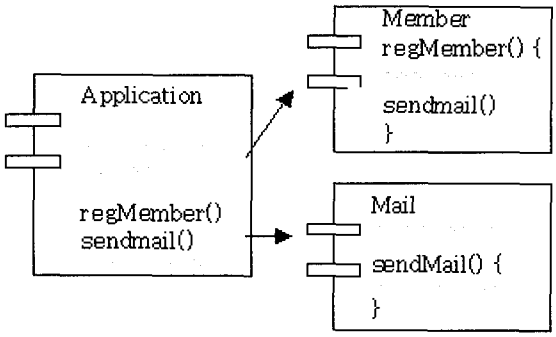


클라이언트에서는 조립된 컴포넌트의 인터페이스만 호출하므로 단순해 진다.

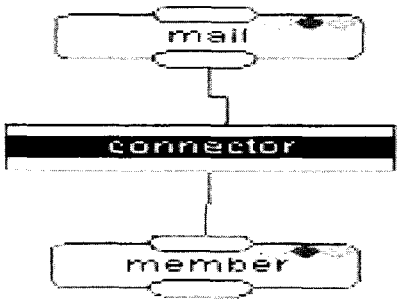
5.5.3 기존 조립방식의 문제점 과 해결점-3

문제점 : Application 컴포넌트를 생성하여 그 컴포넌트가 다른 모든 컴포넌트를 호출

컴포넌트의 재사용이 용이하고, 컴포넌트간의 모든 연관성을 하나의 Application 컴포넌트에 모아 놓아서 비교적 관리하기 쉽다. 하지만 컴포넌트가 많아지고 관계가 복잡해지면 Application 컴포넌트의 개발 및 유지보수가 힘들어 진다.



해결점 : 자동으로 통합된 컴포넌트를 생성해준다



자동으로 컴포넌트를 Generate해주므로 Application컴포넌트의 개발및 유지보수가 필요 없다.

5.5.4 기존 deploy구조와 조립도구구조의 비교

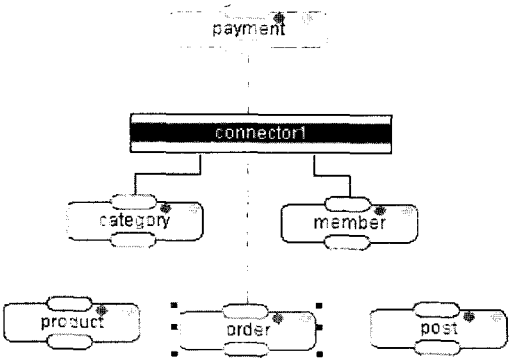
문제점 : EJB Spec에 대한 이해와 Java에 대한 지식이 없으면 구현하기가 힘들다.

```
package shoppingmall;

import javax.ejb.*;
import java.rmi.RemoteException;
import shoppingmall.comp_spec.Category;
import shoppingmall.comp_spec.Payment;
-- 중략 --

public class ShoppingMallBean implements SessionBean
{
   public boolean isMember(java.lang.String m_0,java.lang.String m_1) throws RemoteException
   {
     -- 중략 --
   }

   private SessionContext ctx;
   public void setSessionContext (SessionContext context) throws RemoteException, EJBException {
      ctx = context ;
   }
   public void ejbActivate () throws RemoteException, EJBException {}
   public void ejbPassivate () throws RemoteException, EJBException {}
   public void ejbRemove () throws RemoteException, EJBException {}
   public void ejbCreate (){}
   -- 중략 --
```



해결점: 컴포넌트 조립을 위한 Visual 환경을 제공 하기 때문에 EJB Spec에 대한 이해와 Java에 대한 지식 없이 간단한 툴교육 또는 메뉴얼을 통하여 구현이 가능하다.

5.5.5 기존 deploy와 조립도구의 유지보수 비교

문제점: 유지/보수 차원에서 기존 방식은 프로그램에 대한 전반적인 분석이 선행 되어야 하는 어려움과 해당 수정사항의 흐름을 알아 보기 힘들다.

해결점 : 5.5.4의 그림과 같이 각각의 컴포넌트들의 메시지 흐름을 붉은색으로 표시해주는 기능이 있어 향후 유지/보수 시 쉽게 접근할 수 있다.

5.5.6 기존Deploy와 조립도구를 이용한 초급자 (경력2-3년) 개발 기간 비교

■ 쇼핑몰 프로젝트 개발 기간

- 개발환경 : EJB, J8P, Weblogic
- 소요인원 : 개발자 4명
  프로젝트 매니저 1명
- 프로젝트개발 기간 현황
  - 쇼핑몰 설계 1달
  - J8P 교육 3주(위탁교육:개발자4명)
  - EJB 교육 6주(위탁교육:개발자4명)
  - DB 교육 4주(위탁교육:개발자4명)
  - Weblogic 교육1주(교육인원:개발자1명)
  - J8P 표준 프로그램 Sample 작성 1주
  - EJB 표준 프로그램 Sample 작성 1주
  - 쇼핑몰 소스코딩(테스트포함) 3달
  - 쇼핑몰설계 변경에 의한 소스수정 1달
- 총개발기간 : 약 8개월

■ 조립도구를 이용한 쇼핑몰 프로젝트

- 개발환경 : EJB, J8P, Weblogic
- 소요인원 : 개발자 2명
  프로젝트 매니저 1명
- 프로젝트개발 기간 현황
  - 쇼핑몰 설계 1달
  - J8P 교육 3주(위탁교육 :개발자2명)
  - 컴포넌트조립도구교육1주(자체교육 :개발자2명)
  - Weblogic 교육 1주(교육인원 :개발자1명)
  - 조립도구를 이용한 구현 (테스트포함) 1달
  - 쇼핑몰설계 변경에 의한 수정 2일

5.5.7 기존Deploy와 조립도구를 이용한 초급자 (경력2-3년) 유지보수 기간 비교

■ 쇼핑몰 프로젝트 개발 기간

- 개발환경 : EJB, J8P, Weblogic
- 소요인원 : 개발자 4명
  프로젝트 매니저 1명
- 프로젝트개발 기간 현황
  - 쇼핑몰 설계 1달
  - J8P 교육 3주(위탁교육:개발자4명)
  - EJB 교육 6주(위탁교육:개발자4명)
  - DB 교육 4주(위탁교육:개발자4명)
  - Weblogic 교육1주(교육인원:개발자1명)
  - J8P 표준 프로그램 Sample 작성 1주
  - EJB 표준 프로그램 Sample 작성 1주
  - 쇼핑몰 소스코딩(테스트포함) 3달
  - 쇼핑몰설계 변경에 의한 소스수정 1달
- 총개발기간 : 약 8개월

■ 조립도구를 이용한 쇼핑몰 프로젝트

- 개발환경 : EJB, J8P, Weblogic
- 소요인원 : 개발자 2명
  프로젝트 매니저 1명
- 프로젝트개발 기간 현황
  - 쇼핑몰 설계 1달
  - J8P 교육 3주(위탁교육 :개발자2명)
  - 컴포넌트조립도구교육1주(자체교육 :개발자2명)
  - Weblogic 교육 1주(교육인원 :개발자1명)
  - 조립도구를 이용한 구현 (테스트포함) 1달
  - 쇼핑몰설계 변경에 의한 수정 2일

6. 결 론

SW컴포넌트의 개발은 조립을 전제로 개발되어져야 함에도 불구하고 현재 개발되고 있는 기업이나 기관의 현장은 또 다른 낭비 요소를 가져올 수 있고, 정부 입장에서는 컴포넌트 산업 활성화에 수백억을 투자하고도 효과를 가져오지 못하는 결과를 가져올 수 있다. 재사용 가능한 컴포넌트를 만들도록 지속적으로 유도해야 하는 협회나 정부도 이제는 시장에 맡겨두는 형편이 되었다. 초기에 컴포넌트 산업 활성화 방향을 잘 잡고 아키텍쳐 기반의 컴포넌트 개발에 역점을 두었더라면 품질좋은 재사용 컴포넌트가 많이 생산되어 유통으로 이어질 수 있었으리라 짐작이 된다. 조립이 잘 될 수 있는 컴포넌트를 생성하려면 각 산업 분야의 영역 전체 아키텍쳐가 먼저 만들어지고 그 영역 아키텍쳐의 각 요소 영역별로 컴포넌트가 만들어진다면 조립이 용이할 것이다. 그러나 현재는 Bottom up방식의 컴포넌트 개발은 재사용 컴포넌트의 조립을 더욱 어렵게 만들고 있다. 각 기업

별 특화된 컴포넌트의 개발도 유지보수나 중복 개발비 절감에는 도움이 되겠지만 타 영역으로의 파급효과나 해외 수출의 효과는 매우 떨어질 수 밖에 없을 것이다. 그리고 시스템별로 컴포넌트가 많이 만들어질수록 컴포넌트의 관리와 유지보수가 어려워지고 있다. 컴포넌트 조립도구는 이러한 컴포넌트 유지보수와 재사용, 컴포넌트의 대체성이 뛰어나 향후 컴포넌트 개발에 이 조립도구가 없이 개발되어진다는 것은 현대화된 기계농업과 전통적인 농업의 차이를 나타낼 것이라 본다. 향후 컴포넌트 개발시 조립방법론이 내포된 컴포넌트 개발과 조립도구의 사용에 의한 쉬운 유지보수, 시스템 재개발의 편리성 등으로 이어져 국내외 컴포넌트 산업 발전에 더욱 더 기여할 것을 기대한다. 본 논문에서 간단한 쇼핑몰 시스템을 컴포넌트 조립기반으로 개발함으로써 개발기간의 단축과 비용의 절감을 가져온다는 것을 나타내고 있다. 향후 연구가 더 진행되어 컴포넌트 생성도구와 조립도구의 통합을 통하여 더 효과적인 컴포넌트 개발 지원도구로서의 역할을 담당하게 되고 개발자들의 개발 편의성을 한층 더 증대시킬 수 있으리라 기대한다.

참고문헌

[1] Ralph Keller, Urs Holzle," Implementing Binary Component Adaptation for Java", www.cs.ucsb.edu/oocsb

[2] Oh-Cheon Kwon, Yoo-Hee Choi and Gyu-Sang Shin, " Technique for Adapting EJB Components according to Adaptation Patterns", ISE2001, Las Vegas, USA, 2001

[3] 조진희, 하수정, 김진삼, 박창순, "컴포넌트 기반 시스템 개발 방법론 마르미-III", 프로젝트관리기술논문집, 한국프로젝트관리 기술협회, 제 4권 제 4호, 2001. 9

[4] 권오천, 이우진, 천윤식, 신규상, "CBD 지원 도구의 설계 및 프로토타이핑", 한국정보과학회 정보과학회지, 제 19권 2호, 2001. 2

[5] Rosenblum, D.S. and Natarajan, R.," Supporting architectural concerns in component-interoperability standards", IEE Proceedings- Software, Volume: 147 Issue: 6, pp.215 -223, Dec. 2000

[6] Rational Software Corporation, Rational Unified Process, http://www.rational.com/products/rup/index.jsp

[7] Computer Associates, CBD96, http://www.sterling.com/ cbdedge1/cbd96.htm

[8] CompuWare Corporation, UNIFACE Development Methodology, http://www.compuware.com/

[9] PrinceTonSoftech, Select Perspective, http://www. princetonsoftech.com/index.asp

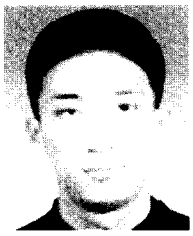
저자약력

정 연 대

1971년 3월~1978년 6월 서강대학교 수학과 졸업
1984년 3월~1988년 2월 서강대학교 경영대학원 졸업
1982년 3월~1984년 2월 KAIST 경영과학과 수료
1978년 6월~1995년 12월 한국과학기술연구원(KIST)부설 시스템공학연구소 선임연구원, 데이터베이스연구실장
1998년 2월~1999년 1월 미국 University of Southern California 초빙연구원
1996년 1월~2000년 4월 한국전자통신연구원(ETRI)컴퓨터.SW 기술연구소 자동화지원도구 개발 실장, 책임연구원
2000년 4월- 현재 (주)N3 SOFT 대표이사
관심분야 : 재사용, UML, CBD, EJB, Product Line, Web Services, MDA 기술 등
이 메 일 : ydchung@n3soft.co.kr

임 진 수

1992년 3월~1999년 2월 연세대학교 수학과 졸업
1999년 11월~2000년 4월 (주)Isoft 근무
2000년 5월- 현재 (주)N3 SOFT 근무
관심분야 : 재사용, UML, CBD, EJB, MDA기술 등
이 메 일 : i1876@n3soft.co.kr



오 연 재

1995년 3월~2000년 8월 충남대학교 물리학 졸업
2001년 1월- 현재 (주)N3 SOFT 근무
관심분야 : 재사용, UML, CBD, EJB, MDA기술 등
이 메 일 : alteration@n3soft.co.kr