아키텍처에 기반한 컴포넌트 조립 시스템의 설계 및 구현 방법과 지원 도구의 개발

(A Method for Architecture-based Design and Implementation of Component Assembly and its Tool Support)

이 승 연 † 권 오 천 †† 신 규 상 †
(Seung-Yun Lee) (Oh-Cheon Kwon) (Gyu-Sang Shin)

요 약 복잡한 응용 프로그램의 빠른 개발과 이의 용이한 유지 보수를 지원하기 위하여 재사용 가능한 컴포넌트 기반 개발(CBD: Component-Based Development) 개념이 확산되고 있다. 이와 관련되어 EJB, COM, CCM 등과 같은 컴포넌트 모델에 대한 연구가 다양하게 진행되고 있으나, 컴포넌트의 궁극적 목표인 재사용 극대화의 입장에서, 아직까지 이미 개발된 컴포넌트들의 유연한 조립은 지원하지 못한다. 이를 해결하기 위해서는 제삼자에 의해 제공된 이질적인 컴포넌트들을 유연하게 재구성 및 조립할 수 있는 상위 레벨의 아키텍처가 필요하며 그러한 아키텍처를 기반으로 구현된 컴포넌트들이 조립되어야 한다. 본 논문은 제삼자에 의해 제공된 이질적인 컴포넌트들을 플러그 앤 플레이 방식으로 유연하게 재구성 및 조립할 수 있도록 아키텍처 기반의 컴포넌트 조립 시스템 설계 및 구현 방법을 제안하고 이를 지원하는 CASE 도구인 Cobalt Assembler를 소개한다.

키워드 : 컴포넌트 조립, 소프트웨어 아키텍처, 아키텍처 기술 언어, 컴포넌트 기반 개발, CASE 도구

Abstract Component-Based Development(CBD) leverages software reusability and diminishes development costs. Various works about component models, such as EJB, COM, and CCM are in progress to support CBD. However, current component models hardly support flexible assembly of pre-built components. To cope with this problem, architecture for component assembly must be defined in the abstract level and the gap between system architecture and its implementation should be diminished in the implementation level. This paper proposes a method for architecture-based design and implementation of component assembly. Architecture is described by the ADL, and the tool, COBALT Assembler, is introduced to support the proposed design and implementation phase of component assembly.

Key words : Component Assembly, Software Architecture, Architecture Description Language (ADL), Component-Based Development, CASE Tool

1. 서 론

복잡한 응용 프로그램을 빠르게 개발하고 이의 유지 보수가 용이하도록 하기 위해 재사용 가능한 컴포넌트 기반 개발(CBD: Component -Based Development) 방법론이 확산되고 있다. 기존 컴포넌트들간의 상호관계를 식별하고 어떻게 재사용할 것인지에 대한 관심이 높아지면서, EJB, COM, CCM 등과 같은 컴포넌트 모델에 대한 연구가 활발히 진행되고 있다[1].

제 삼자에 의해 이미 개발된 컴포넌트들을 조립하여 시스템을 개발할 때에는, 코드의 수정이나 재컴파일 없이 다양한 플랫폼에 원하는 컴포넌트들을 전개하여 사용할 수 있어야 한다. 그러나, 재사용 하고자 하는 컴포넌트들 간의 상호작용이 가능하기 위해서는 직접적인 메소드 호출 로직이 필요하다. EJB 컴포넌트의 경우, 제삼자에 의해 개발된 각각의 컴포넌트들이 상호 작용할 수 있도록 구성되기 위해서는 그들을 사용하는 클라이언트 레벨에서 각각의 메소드를 호출하는 형태로 클라이언트 코드를 작성해야 한다[2]. 이러한 문제를 해결하기 위해서는 컴포넌트들이 서로 정확하게 결합하여

· 본 연구는 2002년 과학기술부 국가지정연구실 사업으로 수행되었음
† 정 회 원 : 한국전자통신연구원 컴퓨터소프트웨어연구소 연구원
coral@etri.re.kr
†† gsshin@etri.re.kr
비 회 원 : 한국전자통신연구원 컴퓨터소프트웨어연구소 팀장
ockwon@etri.re.kr
논문접수 : 2002년 5월 16일
심사완료 : 2003년 4월 9일

작동할 수 있는 기반 구조가 마련되어야 하고, 이를 바탕으로 플러그 앤 플레이(Plug-&-Play) 방식의 유연한 재구성 및 조립이 지원되어야 한다. 즉, 시스템의 설계 단계에서 아키텍처를 기반으로 컴포넌트의 조립구조를 나타낼 수 있어야 하며, 설계된 아키텍처를 바탕으로 시스템 개발에 사용될 구현 컴포넌트들과 연결될 수 있어야 한다[3].

본 논문은 제삼자에 의해 제공된 이질적인 컴포넌트들을 플러그 앤 플레이 방식으로 유연하게 재구성 및 조립할 수 있도록 컴포넌트 조립 시스템의 아키텍처 기반 설계 및 구현 방법을 제안하고 이를 지원하는 CASE 도구인 Cobalt Assembler를 소개한다. 개발하고자 하는 시스템에 대한 잘 정의된 아키텍처가 존재하면, 시스템이 상위 수준에서 어떻게 구성되어 있고 어떠한 기능을 하는지의 방향을 지시해 주므로, 조립하는 시스템이 제공하는 서비스를 추상화하여 표현할 수 있다. 또한, 잘 정의된 아키텍처를 통하여 개발하는 소프트웨어 시스템의 기능을 분석하고 정제 및 검증해 볼 수 있다. 따라서, 아키텍처를 기반으로 컴포넌트들을 조립하면 아키텍처 모델링, 분석, 정제의 과정을 통하여 컴포넌트들이 정확하게 결합하여 작동될 수 있다.

본 논문에서는 시스템 전체의 관점에서 상호작용하는 재사용 대상 컴포넌트들을 연결하여 시스템 아키텍처를 설계하고, 각각의 컴포넌트 관점에서 컴포넌트의 기능 및 행위에 대해 기술하여 컴포넌트 구조를 설계한다. 시스템을 구성하는 컴포넌트의 인터페이스를 정의하고, 커넥터를 통하여 상호 작용하는 컴포넌트들의 인터페이스들을 연결하면 컴포넌트간의 정확한 결합을 지원할 수 있다[4]. 설계된 아키텍처는 C2 스타일[5,6]을 따르는 아키텍처 기술 언어를 이용하여 정형화하며, 기술된 아키텍처 명세에 따라 생성된 컴포넌트 wrapper와 시스템 접속코드는 클라이언트 프로그램과 결합되어 수행될 수 있는 어플리케이션 시스템으로 구현된다.

본 논문의 구성은 다음과 같다. 2장은 아키텍처 기반의 컴포넌트 조립시스템 개발과 관련된 연구에 대해 소개한다. 3장에서는 아키텍처에 기반한 컴포넌트 조립 시스템의 설계 및 구현 방법을 제안하고, 이를 지원하는 도구인 COBALT Assembler를 4장에서 소개한다. 5장에서는 제안한 시스템 설계 및 구현 방법과 개발한 도구를 이용하여 쇼핑몰 시스템의 개발 과정을 보인다. 마지막으로 6장에서는 결론을 맺는다.

2. 관련 연구

기존에 만들어진 컴포넌트들을 재사용 하고자 하는 노력으로 아키텍처 스타일을 이용하여 컴포넌트를 조립하려는 노력이 있어 왔다. D.S.Rosenblum은 C2 아키텍처 스타일에 따라 자바 빈(JavaBeans) 컴포넌트들을 조립하는 방법을 제안하고 자바 빈 컴포넌트들을 조립할 수 있는 아키텍처 모델링 도구인 ARABICA를 언급하였다[7]. 이 논문에서는 C2 wrapping 방법을 사용하여 기존 컴포넌트를 C2 컴포넌트로 매핑하거나, 빈(Bean)의 속성, 메소드, 이벤트를 추출하여 메시지 전달 방식의 인터페이스를 형성하였다. 그러나, 자바 빈 컴포넌트 모델이 서버측 컴포넌트 모델이 아닌 그래픽 사용자 인터페이스 컴포넌트의 재사용을 위한 컴포넌트 모델이라는 점에서 한계가 있다. GUI 컴포넌트의 경우, 비즈니스 로직에 대한 코드를 직접 작성하지 않아도, 정해진 속성, 메소드, 이벤트 편집의 형태로 컴포넌트를 조립할 수 있다. 그러나, 비즈니스 로직을 갖는 컴포넌트들을 조립할 경우에는 개발자가 작성한 로직에 따라 다양한 메소드들이 정의될 수 있고, EJB 컴포넌트의 경우 하나 이상의 빈들로 구성되어, 패키지(Jar) 단위로 이용될 수 있다. 패키지 단위의 컴포넌트를 조립할 때, 이를 구성하는 하나 이상의 빈들이 가지는 기능을 모두 고려하여 메시지 형태로 생성해 주어야 하며 이를 위하여 패키지를 구성하는 빈들의 정보와 이들간의 관계를 정의할 수 있는 편집 과정이 필요하다. 또한, 위 논문에서 언급한 ARABICA는 자바 빈 컴포넌트로 구성된 시스템 아키텍처를 모델링하는 기능은 지원하고 있으나, 이를 정제하여 구현코드를 생성하는 것은 지원하지 않고 있기 때문에 시스템 아키텍처를 구현된 컴포넌트와 연결하기 어렵다.

Zen-Wei Hong은 분산된 환경에서 COTS 컴포넌트들을 통합하기 위한 아키텍처 스타일인 DSIAS 스타일을 제안하고 통합하고자 하는 시스템의 설계에 대한 가이드라인을 제시하였다[8]. DSIAS 스타일은 COTS 컴포넌트들의 재사용성을 높이기 위하여 Wrapping 기술과 개조(Adapting) 기술을 지원하여 다양한 플랫폼에 통합될 수 있도록 지원하며, UI Generator와 Task Coordination Controller의 두 중재자(Coordinator)를 두어 사용자의 요구사항을 받아들이고, 이를 수행하기 위하여 COTS 컴포넌트들이 서로 상호 작용할 수 있도록 지원한다. 그러나, 이 논문은 설계된 시스템 아키텍처와 COTS 컴포넌트들과의 연결에 관한 언급이 없어 설계를 바탕으로 실제 컴포넌트들이 어떻게 연결되어 구현되는지에 대한 가이드라인을 제시해주지 못한다. 또한 제안한 아키텍처 스타일을 아키텍처 기술 언어로 정형화시켜 표현하고 있지 않아서, 정확하고 엄밀하게 아키텍처를 모델링하고, 아키텍처에 기반하여 시스템을 분석, 정제, 검증하는데 용이하지 않다.

개발된 컴포넌트를 재사용하여 새로운 시스템을 만들기 위해서는 시스템 개발의 전체적인 조립 프로세스가

필요하다. 본 논문은 비즈니스 로직을 가지는 서버측 컴포넌트들의 조립을 지원하기 위하여, 아키텍처 기반의 시스템 설계와 구현방법을 제안하고 EJB 컴포넌트를 제안한 방법으로 조립하는 것을 지원하는 COBALT Assembler 조립도구를 소개한다. 제안하는 시스템의 아키텍처 설계는 C2 스타일을 따르는 아키텍처 기술 언어로 정형화시켜 표현하며, 기술된 아키텍처 명세는 제공된 컴포넌트들의 구현과 연계되어 컴포넌트 조립을 통한 시스템 개발에 이용된다.

3. 아키텍처 기반의 컴포넌트 조립 시스템 설계 및 구현방법

본 장에서는 재사용 가능한 컴포넌트를 조립하여 새로운 어플리케이션 시스템을 아키텍처를 기반으로 개발하는 방법을 제안한다. 그림 1은 조립 시스템을 개발하는 전체 프로세스를 나타낸 것이다.

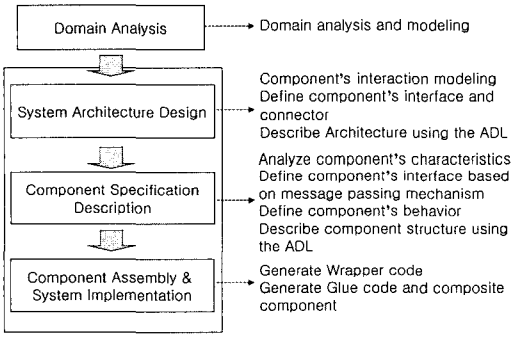

그림 1 컴포넌트 조립 시스템의 개발 프로세스

제공된 컴포넌트들을 이용하여 어플리케이션 시스템을 개발하기 위해서는 도메인에 대한 이해와 사용자의 요구사항에 대한 분석이 선행되어야 한다. 문제 영역 분석 단계에서는 영역 분석 및 모델링 과정을 통하여, 시스템 조립에 이용될 컴포넌트들을 식별한다. 식별된 컴포넌트들은 어플리케이션 시스템 아키텍처 설계에 이용된다. 아키텍처의 설계는 조립 어플리케이션 시스템의 구조와 컴포넌트간의 상호관계를 나타내는 시스템 아키텍처 설계와 식별된 컴포넌트들의 인터페이스를 설계하는 컴포넌트 명세 기술의 두 단계로 나뉘어 진다. 식별된 컴포넌트와 개발하는 시스템의 아키텍처는 아키텍처 기술 언어(ADL)로 명세화되며, 기술된 아키텍처 명세는 실제 조립 어플리케이션 시스템을 구현하는데 필요한 Wrapper 코드와 접속 코드를 생성하는데 이용된다.

3.1절부터 3.3절에서는 영역 분석 및 모델링을 통하여 식별된 컴포넌트들을 이용하여 시스템의 아키텍처와 컴포넌트 구조를 설계하는 방법 및 이를 이용한 구현 시스템의 개발방법을 자세히 설명한다.

3.1 시스템 아키텍처의 설계

그림 2는 제공된 컴포넌트들을 이용하여 개발하는 시스템의 아키텍처를 설계하는 과정을 나타낸 것이다. 문제 영역 분석 단계를 통하여 개발하는 영역에 대한 이해와 사용자의 요구사항에 대한 분석을 하며, 개발에 이용될 컴포넌트들을 식별할 수 있다. 문제 영역 분석 단계를 거쳐 나온 결과를 이용하여 개발하는 어플리케이션 시스템의 전체 아키텍처를 설계하는데, 각각의 단계는 다음과 같다.

1) 컴포넌트간의 행위 모델링(Modeling components' interaction behavior): 식별된 컴포넌트를 이용하여 시스템의 요구사항을 만족하는 컴포넌트간의 행위를 모델링한다. 컴포넌트 명세서를 통하여 컴포넌트가 제공하는 기능들 중 이용될 기능을 정의하고 요구사항 분석서에서 정의된 사용자 시나리오를 기반으로 이들간의 상호작용 흐름의 순차도를 작성한다.

2) 시스템의 중재자 역할을 하는 커넥터 정의(Define a connector as a coordinator): 각각의 컴포넌트에서 이용할 기능들은 인터페이스를 통하여 접근하도록 설계되어야 컴포넌트들의 유연한 재구성을 지원할 수 있다. 컴포넌트간의 상호작용 모델링을 통하여 기술된 Workflow를 수행하기 위하여 컴포넌트의 인터페이스를 다양하게 정의할 수 있으며, 컴포넌트의 다양한 인터페이스를 관리하기 위하여 컴포넌트와 컴포넌트의 연결 시 커넥터를 정의하여 시스템의 통로 역할을 담당한다.

3) 컴포넌트와 커넥터를 이용한 시스템 아키텍처 설계(Design system architecture): 식별된 컴포넌트들과 정의된 커넥터들을 이용하여 시스템 전체 아키텍처를 설계한다. 설계된 아키텍처를 검증하기 위하여 정형화된 언어로 명세가 필요하며, 본 논문에서는 C2 아키텍처

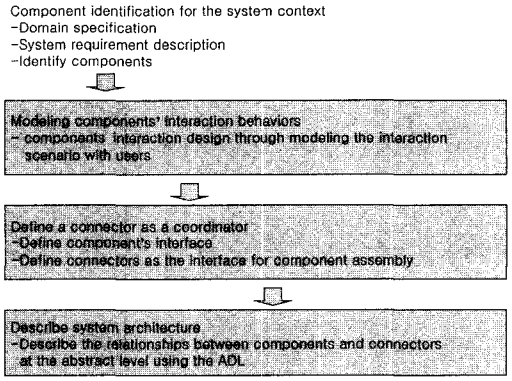

그림 2 시스템 아키텍처의 설계 방법

스타일을 지원하는 자바 스타일의 ADL을 이용하여 컴포넌트와 커넥터의 관계를 기술한다. 시스템 아키텍처를 구성하는 컴포넌트, 커넥터, 그리고 이들간의 *Binding* 관계를 통하여 정의된다[9].

시스템 전체에 대한 아키텍처는 구현되어진 컴포넌트들의 상호관계를 정의함으로써 복잡한 개발 영역을 추상화시켜 표현하고, 이 때 각각의 컴포넌트가 갖는 인터페이스를 커넥터가 관리하도록 함으로써 컴포넌트의 독립성을 높여 플러그 앤 플레이 방식의 조립을 가능하게 한다. 또한, ADL을 이용하여 정확하고 엄밀하게 아키텍처를 모델링함으로써 아키텍처 기반의 시스템 분석, 정제, 및 검증을 할 수 있다.

3.2 컴포넌트 명세 기술

컴포넌트 조립을 통하여 시스템을 개발하기 위해서는 메소드 호출 방식을 메시지 전달 방식으로 수정함으로써 컴포넌트의 상호작용을 유연하게 표현할 수 있다. 시스템 아키텍처의 설계가 플러그 앤 플레이 방식의 컴포넌트 조립을 지원하기 위한 기반을 마련한다면, 컴포넌트 명세 기술은 실제 개발하는 시스템의 요구사항을 만족시키기 위해 필요한 컴포넌트의 기능과 속성을 포트 기반의 메시지 전달 방식으로 추상화시켜 표현함으로써, 기존 컴포넌트의 코드 수정 없이 컴포넌트간의 상호 관계를 명확하게 표현하고 이들을 조립할 수 있는 기반을 마련해 준다. 컴포넌트 명세 기술 단계에서는 상위 레벨에서 구현과 관련 있는 컴포넌트의 메시지 포트, 인터페이스, 메소드와 행위를 기술한다[10].

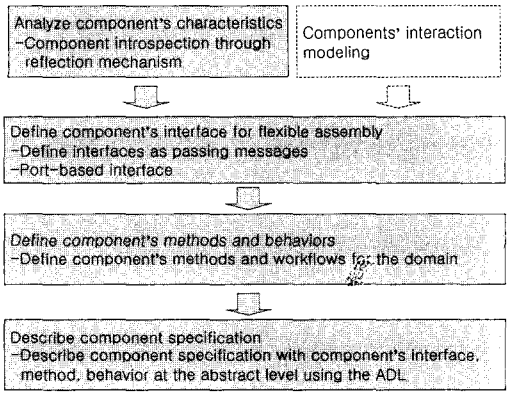

그림 3 컴포넌트 명세 방법

그림 3은 컴포넌트 각각을 명세하는 방법을 나타낸 것으로 각 단계에 대해 살펴보면 다음과 같다.

1) 컴포넌트 정보 분석(Analyze component's characteristics): 식별된 컴포넌트에 대하여 클래스 정보 및 제공하는 기능에 대한 정보를 알아내는 단계로 reflection 기능을 이용하여 컴포넌트 정보를 알아낸다[11,12].

2) 유연한 조립을 위한 컴포넌트 인터페이스의 정의(Define component's interface for flexible assembly): 컴포넌트 정보 분석을 통하여 얻어진 정보와 시스템 아키텍처 설계시 사용되었던 컴포넌트 상호 작용 모델링 결과를 이용하여 개발하는 시스템 영역에서 컴포넌트가 반드시 제공해야 하는 기능들을 정의하고 이를 메시지 전달 방식의 인터페이스로 표현한다. 컴포넌트는 포트를 통하여 메시지를 전달하게 되는데, 이 때 메시지는 <id>(<parameter_list>)와 같이 메시지 이름과 파라미터 리스트의 구성을 가지며, 컴포넌트의 포트 부분에 정의된다.

3) 컴포넌트의 메소드 및 이를 이용한 행위 정의(Define component's methods and behaviors): 메시지 전달 방식의 인터페이스 정의 후, 컴포넌트의 메소드 리스트 중에서 개발되는 시스템에 사용될 메소드를 선택하고, 정의된 인터페이스와 메소드의 수행흐름을 컴포넌트의 행위로 정의한다. 정의되는 행위는 개발하는 영역에 의존적으로 하나의 컴포넌트가 여러 도메인에 사용될 때 다른 행위를 정의할 수 있다. 컴포넌트의 동작 시작, 실행, 또는 종료시 행해지는 행위를 메소드 호출과 포트를 통한 메시지 송신으로 표현한다.

4) 컴포넌트 명세 기술(Describe component specification): 앞에서 정의한 컴포넌트의 인터페이스와 메소드, 행위를 시스템 아키텍처 설계시와 마찬가지로 C2 스타일의 ADL 언어로 기술한다.

컴포넌트 명세는 개발하는 시스템 아키텍처와 식별된 재사용 가능한 컴포넌트들을 연결시켜주는 매개체 역할을 하는 것으로, 작성된 컴포넌트 명세 파일은 3.3절의 조립 시스템을 구현할 때 각각의 컴포넌트들을 조립 가능한 형태로 변환시키는데 사용된다.

3.3 조립 시스템의 구현

3.1절과 3.2절에서 조립될 어플리케이션 시스템의 전체 아키텍처를 설계하고 컴포넌트를 명세화하는 방법을 설명하였다. 기술된 아키텍처 명세는 실제 컴포넌트들을 조립하여 시스템을 구축하는 단계에서 필요하다. 컴포넌트 명세를 이용하여 컴포넌트의 인터페이스를 메시지 전달 방식으로 Wrapping하여 조립이 용이하게 하며, 시스템 아키텍처 명세를 고려하여 컴포넌트들을 하나의 시스템으로 구성해 주는 접속 코드와 합성 컴포넌트를 생성한다[13].

그림 4는 설계된 아키텍처 명세를 바탕으로 어플리케이션 시스템을 조립하는 방법을 나타낸 것이다. 설계한 아키텍처와 어플리케이션 시스템 구현과의 연결은 C2 아키텍처의 구조를 따르는 프레임워크에 기반한다. 이는

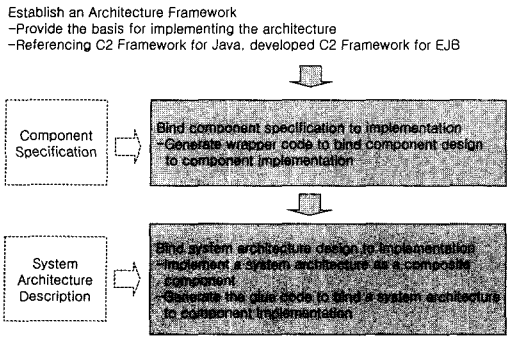

그림 4 조립 시스템의 구현 방법

시스템 아키텍처 구현에 필요한 컴포넌트, 커넥터 등의 정의와 컴포넌트의 인터페이스, 메소드, 행위 등에 대한 구조를 시스템 구현에 사용될 언어로 제공해준다. 구현된 프레임워크를 바탕으로 조립시스템을 구현하는 단계는 다음과 같다.

1) 컴포넌트 명세와 구현 컴포넌트의 연결(Bind component specification to implementation): 컴포넌트 명세에서 기술한 컴포넌트의 인터페이스, 메소드, 행위들은 실제 제공된 컴포넌트와 연결되어 기존의 컴포넌트들을 플러그 앤 플레이 방식으로 조립할 수 있도록 지원해야 한다. 이를 위하여 각각의 컴포넌트는 메시지 전달 방식의 인터페이스를 갖도록 생성된 Wrapper 코드를 통하여 상호 작용한다. Wrapper 코드는 컴포넌트 명세와 프레임워크를 이용하여 생성되며 컴포넌트의 구현 코드를 설계된 명세에 맞게 Wrapping하는 역할을 한다.

2) 시스템 아키텍처와 구현 컴포넌트들의 연결(Bind system architecture design to implementation): 각각의 컴포넌트가 유연한 재사용을 지원하도록 Wrapping된 후에는 개발하는 시스템의 아키텍처에 맞게 컴포넌트들을 연결하여 새로운 합성 컴포넌트를 생성한다. 이때, 접속 코드는 제공된 프레임워크를 이용하여 개발하는 시스템의 아키텍처 명세와 이를 구성하는 컴포넌트들을 연결시키는 역할, 즉 합성 컴포넌트와 시스템 아키텍처를 연결하는 역할을 한다.

설계단계에서 기술된 아키텍처 명세는 컴포넌트들의 독립성을 유지시키면서 플러그 앤 플레이 방식의 손쉬운 조립을 지원하기 위해 작성되었으며, 이를 Wrapper 코드와 접속 코드를 이용하여 시스템 구현에 반영함으로써 재사용 가능한 컴포넌트들을 조립하여 새로운 어플리케이션 시스템을 개발하도록 지원한다. 또한, 설계된 시스템 아키텍처를 하나의 합성 컴포넌트로 생성함으로써 새로운 재사용 가능한 컴포넌트로 이용될 수 있도록 지원한다.

4. COBALT Assembler: 컴포넌트 조립 시스템 개발을 지원하는 CASE 도구

3장에서 제안한 컴포넌트 조립 시스템 설계 및 구현 방법을 지원하기 위하여 Cobalt Assembler라는 컴포넌트 조립지원도구를 개발하였다. 그림 5는 그림 1의 시스템 설계 및 구현을 위하여 지원해야 하는 기능과 그 기능을 수행하는 도구의 모듈들을 나타낸 것이다.

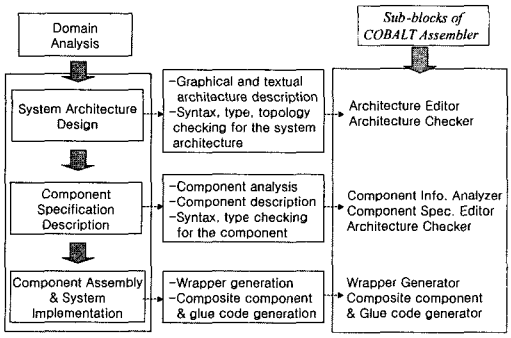

그림 5 조립 시스템 설계 및 구현을 위한 CASE 도구의 지원 기능 및 모듈

문제 영역 분석 단계에서 식별된 컴포넌트들을 토대로 아키텍처 편집기를 이용하여 시스템의 구조를 설계한다. 컴포넌트들의 관계에 대한 정보를 이용하여 컴포넌트와 컴포넌트 사이에 커넥터를 두어 중재자 역할을 하게 하며, 이들간의 연결은 아키텍처 명세 파일로 저장된다. 아키텍처 편집기를 이용하여 작성된 아키텍처 명세는 아키텍처 검사기를 이용하여 올바르게 연결되었는지 검사하는 과정을 거치게 된다.

각각의 컴포넌트에 대하여 컴포넌트 정보 분석기를 이용하여 재사용하는 컴포넌트의 기능을 분석하여 보여주고, 분석된 정보는 컴포넌트 명세를 기술하는데 이용된다. 컴포넌트 명세는 플러그 앤 플레이 방식의 시스템 조립을 지원하기 위해 메시지 전달 방식의 인터페이스를 가져야 하고 메시지 및 메소드의 처리 과정을 컴포넌트의 행위로 정의한다. 이러한 작업은 컴포넌트 명세 편집기를 통하여 수행되며 그 결과는 컴포넌트 명세 파일로 저장된다. 저장된 컴포넌트 명세는 아키텍처 검사기를 통하여 올바르게 작성되었는지 검사한다.

COBALT Assembler에서는 아키텍처와 컴포넌트를 그림 6과 같은 ADL의 구조를 이용하여 정의한다. ADL은 아키텍처를 기술하는 ADN(Architecture Description Notation)과 컴포넌트 명세를 기술하는 IDN (Interface Definition Notation)으로 나누어진다. ADN에서는 아키텍처의 구성요소인 컴포넌트와 커넥터, 이들간

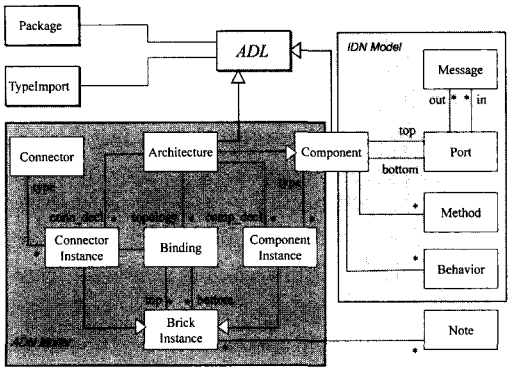

그림 6 아키텍처를 정형화하는 ADL의 구조

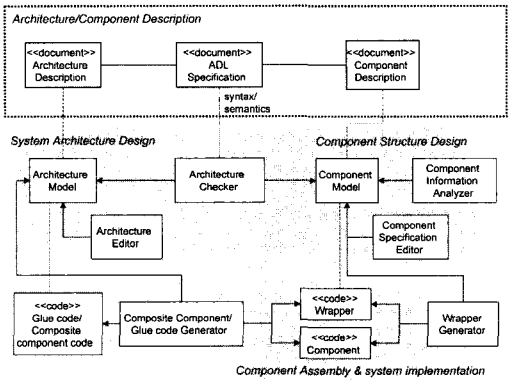

그림 7 COBALT Assembler의 구조 및 단위 모듈과 산출물간의 관계

의 관계를 표현하기 위한 모델을 정의하고, IDN에서는 컴포넌트의 메시지, 메소드, 행위 등을 기술하기 위한 모델을 정의한다.

작성된 아키텍처 명세와 컴포넌트 명세를 이용하여 새로운 어플리케이션 시스템을 개발한다. Wrapper 생성기는 작성된 컴포넌트 명세와 컴포넌트가 가지는 메소드를 이용하여 메시지 전달 방식의 컴포넌트 Wrapper 코드를 생성한다. 접속코드 생성기는 wrapping된 컴포넌트들을 명세된 아키텍처로 연결하는 접속코드를 생성한다. 합성 컴포넌트 생성기는 접속코드에 의해 연결된 컴포넌트들을 단일한 합성 컴포넌트로 생성한다.

그림 7은 COBALT Assembler의 구조 및 단위 모듈과 산출물간의 관계를 나타낸 것이다. 컴포넌트들을 이용하여 어플리케이션 시스템을 개발하는데 생성되는 산출물은 시스템 아키텍처를 기술하는 아키텍처 명세, 컴포넌트의 인터페이스 및 행위를 나타내는 컴포넌트 명세, 컴포넌트 조립에 쓰이는 Wrapper 코드와 접속 코드 등이 있다. 아키텍처 편집기를 이용하여 작성된 아키텍처 모델은 아키텍처 명세 파일로 저장되며, 이는 ADL 파서/검사기를 통하여 검사된다. 아키텍처를 구성하는 각각의 컴포넌트는 컴포넌트 편집기를 통하여 컴포넌트 모델을 생성하며 이때 컴포넌트 정보 분석기에서 분석된 컴포넌트 정보를 이용하게 된다. 생성된 컴포넌트 모델은 컴포넌트 명세 파일로 저장되며 저장된 명세 파일과 제공된 컴포넌트를 이용하여 컴포넌트 Wrapper 코드를 생성하고 아키텍처와 연결해 주는 접속 코드와 합성 컴포넌트 코드를 생성한다.

COBALT Assembler는 3장에서 제안한 컴포넌트 조립 시스템의 설계 및 구현 방법을 기반으로 EJB 컴포넌트들의 조립을 지원한다. 본 도구는 EJB 컴포넌트들이 플러그 앤 플레이 방식으로 손쉽게 조립될 수 있도록 시각적 편집을 지원한다. 개발하는 시스템의 아키텍처와 각 컴포넌트의 명세를 기술하면, COBALT Assembler는 자동으로 Wrapper 코드와 접속 코드를 생성하여 시스템의 설계와 컴포넌트 구현을 연결해 준다. 이는 제 삼자에 의해 개발된 컴포넌트들을 이용하여 하나의 시스템을 구축하고 이들간의 상호 작용할 수 쉽게 정의하는데 도움을 준다.

5. 아키텍처 기반의 컴포넌트 조립 시스템 설계 및 구현방법의 쇼핑몰 적용 예

본 장에서는 COBALT Assembler를 이용하여 EJB 컴포넌트들로 구성된 간단한 쇼핑몰 사이트를 개발해 보고, 제안한 방법을 검증한다.

문제 영역 분석 단계의 결과로 개발하는 쇼핑몰 사이트의 영역 분석서와 요구사항 분석서가 존재하며 이를 통하여 다음과 같은 기능을 수행하여야 함을 가정한다.

1) 사용자는 쇼핑몰 사이트에 접속하여 사용자 인증을 받아야 한다.

2) 사용자는 쇼핑몰 사이트에 접속하여 물건을 구매할 수 있다. 이때 원하는 품목의 재고가 없으면 구매할 수 없다.

1)과 2)를 수행하기 위하여 식별된 컴포넌트는 사용자를 인증하는 Member 컴포넌트, 주문을 처리하는 Order 컴포넌트, 그리고 재고를 관리하는 Stock 컴포넌트이다.

5.1 COBALT Assembler를 이용한 시스템 아키텍처 설계

시스템의 아키텍처를 설계하기 위하여, 쇼핑몰 시스템이 구현해야 하는 요구사항 중 주문과 관련된 처리가 식별된 컴포넌트들을 이용하여 어떻게 진행될 것인지 모델링한다. Order 컴포넌트에 사용자의 주문요청이 들어오면, Order 컴포넌트는 Stock 컴포넌트의 재고 처리 기능을 이용하여 재고가 있는지 없는지를 알아내며, Member 컴포넌트의 사용자 인증 기능을 이용하여 고객이 쇼핑몰 사이트의 가입자인지를 확인해야 한다. 즉, 식

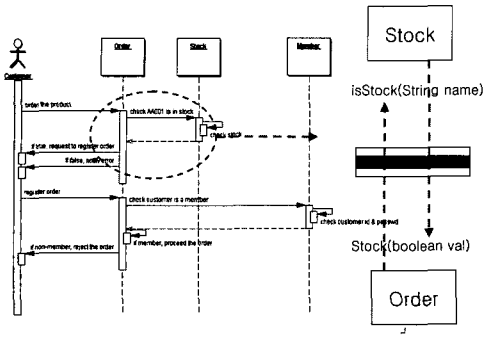

그림 8 Stock 컴포넌트와 Order 컴포넌트간의 인터페이스 정의

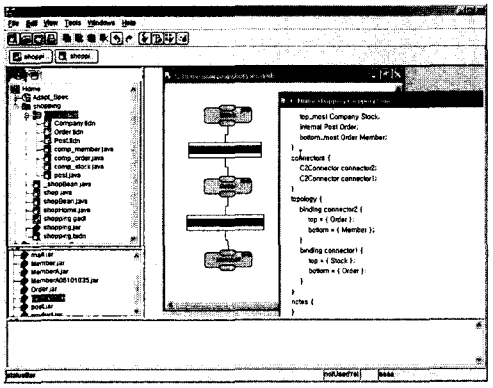

그림 9 COBALT Assembler에서 표현한 아키텍처 명세

별된 세 개의 컴포넌트들이 조립되어 수행되어야 한다.

정의된 컴포넌트 행위 모델링을 통하여 각 컴포넌트가 가져야 하는 인터페이스를 정의할 수 있으며, 메시지 전달의 통로 역할을 하는 커넥터를 통하여 호출된다. 그림 8은 컴포넌트 행위 모델링을 통하여 Stock 컴포넌트와 Order 컴포넌트 사이의 인터페이스를 정의한 것이며, COBALT Assembler에서 컴포넌트와 커넥터를 이용하여 구성한 아키텍처 다이어그램과 이를 표현한 아키텍처 명세는 그림 9와 같다.

5.2 COBALT Assembler를 이용한 컴포넌트 명세 기술

개발하는 시스템의 전체적인 아키텍처를 그림 9와 같이 설계한 후, 아키텍처를 구성하는 컴포넌트들의 명세를 기술한다. 이 때 Cobalt Assembler의 컴포넌트 정보 분석기를 통하여 컴포넌트의 정보를 얻어오며, 이를 이용하여 조립을 지원하는 컴포넌트 명세를 작성한다.

컴포넌트는 두개의 포트를 가지며, 각각의 포트에 내보내는 메시지와 들어오는 메시지를 정의하여, 외부 컴포넌트가 접근할 수 있는 인터페이스를 정의한다. 컴포넌트 정보 분석기를 통하여 필요한 컴포넌트의 메소드

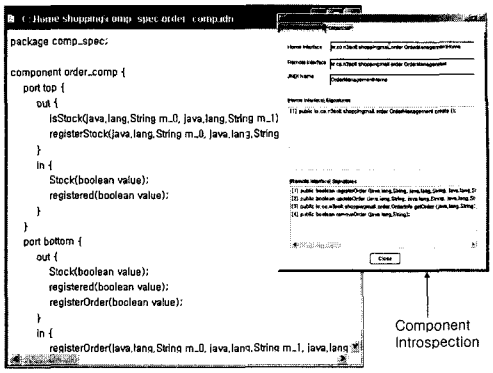

그림 10 COBALT Assembler에서 보여주는 Order 컴포넌트의 정보 및 작성된 명세

를 선택한 후, 받아들이는 메시지에 따라 수행하는 컴포넌트 행위를 정의한다. 그림 10은 COBALT Assembler에서 보여주는 Order 컴포넌트의 메소드 정보 및 ADL로 작성된 명세이다.

5.3 COBALT Assembler를 이용한 Shopping mall 조립 시스템의 구현

5.1절과 5.2절에서 설계된 시스템 아키텍처와 컴포넌트 명세는 쇼핑몰 시스템을 구현하기 위한 컴포넌트 Wrapper 코드와 접속 코드를 생성하는데 이용된다. 기존에 개발한 Order, Stock, Member 컴포넌트는 메시지 전달 방식을 따르고 있지 않기 때문에 작성된 컴포넌트 명세와 기존 컴포넌트를 연결해주는 Wrapper 코드와 아키텍처 명세와 연결되어 전체 시스템을 조립해 주는 접속코드가 필요하다. 또한, 설계된 시스템 아키텍처를 하나의 단일 컴포넌트로 재사용할 수 있게 하기 위하여, 합성 컴포넌트를 생성한다. 그림 11은 Order 컴포넌트의 명세를 이용하여 COBALT Assembler에서 생성한 Wrapper 코드와 컴포넌트들을 연결시켜주는 접속 코드를 나타낸 것이다.

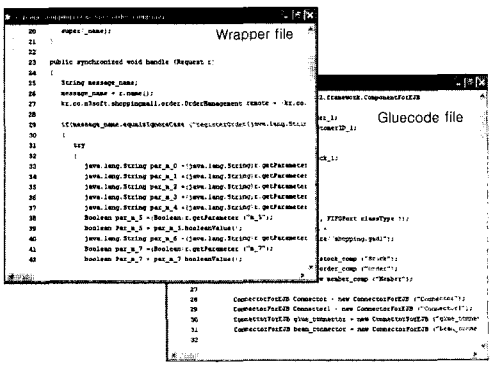

그림 11 COBALT Assembler에서 생성한 wrapper 코드와 접속코드

6. 결 론

본 논문은 CBD의 핵심인 컴포넌트의 재사용성을 높이기 위하여, 제삼자에 의해 제공된 이질적인 컴포넌트들을 플러그 앤 플레이 방식으로 유연하게 조립하여 새로운 어플리케이션 시스템을 개발하는 아키텍처 기반의 설계와 구현 방법을 제안하였다. 또한, 이를 지원하는 도구로 COBALT Assembler를 개발하였으며, 이를 간단한 쇼핑몰 예제에 적용해 보았다.

아키텍처를 기반으로 기존에 개발된 컴포넌트 모델을 연결하여 조립하면 시스템이 상위 수준에서 어떻게 구성되어 있고 어떠한 기능을 하는지의 방향을 지시해 주므로, 제공하는 서비스를 추상화하여 조립하는 방식으로 시스템을 표현할 수 있다. 또한, 잘 정의된 아키텍처를 통하여 개발하는 소프트웨어 시스템의 기능을 분석하고 정제 및 검증해 볼 수 있다. 본 논문은 이러한 장점을 반영하기 위하여, 아키텍처를 기반으로 한 시스템 조립을 유도하였으며, 컴포넌트의 유연한 조립을 위하여 컴포넌트의 아키텍처를 메시지 전달 방식으로 표현하였다. 제안한 방법은 COBALT Assembler를 이용한 사례연구를 통하여 컴포넌트 조립시 아키텍처를 기반으로 설계하고 구현하면 컴포넌트의 대체 및 추가 등의 시스템 유지 보수가 쉽게 처리될 수 있음을 확인하였다.

향후, 개발하는 시스템의 아키텍처와 이를 구성하는 컴포넌트들의 명세를 확장하여야 한다. 컴포넌트 명세에 추가적으로 트랜잭션의 처리, 메시지 흐름의 제어를 위한 상태 정보 및 선/후행 조건 정보 등에 대한 명세가 기술되어야 하며, 컴포넌트의 동적 변화에 대응할 수 있는 아키텍처도 기술될 수 있어야 한다. 아키텍처 명세의 확장과 더불어, 이를 반영하는 COBALT Assembler의 지속적인 지원이 필요하다. 확장된 명세를 지원하는 Wrapper 코드와 접속 코드를 자동 생성할 수 있어야 하며, 앞으로, EJB 컴포넌트 뿐 만 아니라 CORBA, .NET과 같은 다양한 컴포넌트 플랫폼에 적용될 수 있는 조립 지원 도구로 확장되어야 할 것이다. 이와 별도로, COBALT Assembler는 현재 프로토타입이 개발된 컴포넌트 생성 지원 도구(Cobalt Constructor)와 통합되어 컴포넌트 개발 및 조립의 전 단계를 지원할 것이다.

참 고 문 헌

[1] Andersson, J., Johnson, P., "Architectural integration styles for large-scale enterprise software systems," Proceedings of 5th International Enterprise Distributed Object Computing Conference, pp 224-236, 2001.

[2] Felix B., Len B., Charles B., Santiago C.D., Fred L., John R., Robert S. and Kurt W., "Technical Concepts of Component-Based Software Engineering," Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 2000.

[3] Moreira, R.S., Blair, G.S., Carrapatoso, E, "A reflective component-based and architecture aware framework to manage architecture composition," Proceedings of 3rd International Symposium on Distributed Objects and Applications, pp 187-196, 2001.

[4] Desmond D'Souza, D.F., and Alan.C.Wills, Object, Components, and Frameworks with UML: The Catalysis Approach, Addison-Wesley, Reading, MA, 1999.

[5] N. Medvidovic, D. S. Rosenblum, and R. N. Taylor, A Language and Environ ment for Architecture-Based Software Development, Proceedings of the 21st International Conference on Software Engineering (ICSE 21), Los Angeles, CA, pp 44-53, May, 1999.

[6] Nenad Medvidovic, Peyman Oreizy, and Richard N. Taylor, Reuse of off-the-shelf components in C2-style architectures," Proceedings of the Symposium on Software Reusability, pp 190-198, Boston, MA, May, 1997.

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

[8] Zen-Wei Hong, Jim-Min Lin, Jiau, H.C., De-Sheng Chen, DSIAS: a software architectural style for distributed software integration systems, 25th Annual International Computer Software and Applications Conference, 2001, pp 291-296, 2001.

[9] 신동익, 노성환, 최재각, 전태웅, 이승연, 권오천, 신규상, "도메인 아키텍처 기반의 CBD 지원을 위한 ADL의 정의와 이의 지원도구 개발", 정보처리학회 논문지 D(특집호), 2001.12

[10] Richard N. Taylor, Nenad Medvidovic, and etc., A component and message based architectural style for GUI software, In IEEE Transactions on Software Engineering, Volume 22, No 6, pp 390-406, June 1996.

[11] Blair, G., Blair, L., Issarny, V., Tuna, P., and Zarras, A., The role of software architecture in constraining adaptation in component-based middleware platform, Middleware 2000.

[12] Costa, F., Blair, G., and Coulson, G., Experiments with reflective middleware, ECOOP'98, Brussels, Belguim, 1998.

[13] 최유희, 권오천, 신규상, "C2 스타일을 이용한 EJB 컴포넌트의 합성 방법", 정보처리학회 논문지 D(특집호), pp 771-780, 2001.12

이 승 연

1999년 서강대학교 컴퓨터학 공학사 2001년 서강대학교 컴퓨터학 공학석사 2001년~현재 한국전자통신연구원 컴퓨터소프트웨어연구소. 임베디드 S/W 기술센터 연구원. 관심분야는 소프트웨어 아키텍처, 컴포넌트 개발 방법론, 컴포넌트 조립, MDA

권 오 천

1994년 영국 Teesside 대학교 대학원 S/W공학과 공학석사. 1998년 영국 Durham 대학교 대학원 전산과학과 공학박사. 1991년 미국 RTP IBM 연구소 객원연구원. 1993년 시스템공학연구소 선임연구원. 2000년 한국전자통신연구원 책임연구원. 현재 ETRI 컴퓨터소프트웨어연구소. 공간정보기술센터 공간정보기반기술연구팀 팀장. 관심분야는 Reuse, CBD, MDA, AOP, Web Services, Product Line

신 규 상

1981년 성균관대학교 통계학과 학사 1983년 서울대학교 대학원 계산통계학과 이학석사. 2001년 충남대학교 대학원 컴퓨터과학과 이학 박사. 1983년 시스템공학연구소 연구원. 1987년 시스템공학연구소 선임연구원. 1997년 한국전자통신연구원 책임연구원. 현재 ETRI 컴퓨터소프트웨어연구소. 임베디드 S/W 기술센터 책임 연구원. 관심분야는 CASE, S/W 컴포넌트 기술, MDD, 웹 서비스 기술, 멀티미디어 시스템