CORBA 환경에서 멀티미디어 컴퍼넌트 관리 시스템을 통한 프레임워크 구축

김행곤†

요 약

프레임워크는 추상화된 클래스들의 집합 또는 특정 도메인에서 재사용가능한 설계를 구성하는 상호연관된 클래스들의 집합으로 도메인에 속한 애플리케이션들간의 공통된 아키텍쳐를 정의한다. 프레임워크를 재사용함으로 개발자들은 클래스 코드뿐만 아니라 도메인에 대한 폭넓은 도메인 지식을 재사용할 수 있다.

따라서 본 논문에서는 프레임워크 재사용을 위해 컴퍼넌트기반의 방법론(COM : Component-Oriented Methodology)을 제시하고 프레임워크 구축환경을 개발한다. 즉, 컴퍼넌트기반의 소프트웨어 생성을 위해 실세계의 도메인 지식을 입력으로 도메인을 분석하고 분석된 정보를 통해 hotspot을 식별하여 사용자와 개발자의 추가정보를 첨가하는 재설계 과정을 거친다. 이후 도메인에 따라 도메인 프레임워크와 애플리케이션 프레임워크를 생성한다. 이러한 컴퍼넌트 지향 방법론은 내부적으로는 컴퍼넌트/패턴 라이브러리 저장소를 통해 정보를 검색, 이해하여 추출하거나 혹은 합성으로 얻어진 정보는 각각 컴퍼넌트와 패턴에 대한 정보로 분류되고 이것은 재설계시 추가정보로 사용되어진다. 이로 인해 사용자는 멀티미디어 컴퍼넌트를 이용해 자신의 애플리케이션에 쉽게 적용가능한 개발 환경 즉, 본 논문에서는 CORBA(Common Object Request Broker Architecture) 환경하에 컴퍼넌트/패턴 관리 시스템(Component Pattern Management System : CPMS)을 통해 멀티미디어 컴퍼넌트를 추가, 갱신, 삭제하는 기반 환경(infrastructure environment)을 구축함으로 재사용성, 사용의 용이성과 이식성을 가질 수 있다.

Framework Construction with Multimedia Component Management System on CORBA

Haeng-Kon Kim†

ABSTRACT

Framework is the set of interrelated classes, constructing reusable design in specific domain or set of abstracted classes, and defines common architecture among applications included in domain. Developers can reuse not only class code but also wide range of knowledge on domain by reusing framework.

In this papers, we present COM(Component-Oriented Methodology) for the reuse of framework, and develop construction environment for framework and domain development. That is, domain is analyzed by input of domain knowledge on real world to create software based on component, and hotspot is identified through analyzed information, and redesigned(refactoring) by putting additional information on users and developers. After that, I will create domain framework and application framework depending on domain. In this Component-oriented methodology, information is searched, understood and extracted or composite through component/pattern library storage internally. Then this information is classified into the information on component and pattern respectively, and used as additional information in redesigning. With this, developer can obtain reusability, easiness and portability by constructing infrastructure environment that allows to register, update and delete component through Component Pattern Management System(CPMS) under the development environment which can be easily applied to his own application using multimedia component, in this thesis, CORBA(Common Object Request Broker Architecture) environment.

† 대구효성가톨릭대학교 컴퓨터 공학과 부교수

1. 서 론

소프트웨어 재사용이란 새로운 소프트웨어 생산과 유지보수에 있어서 기존에 개발되어 있는 소프트웨어로부터 표준화되어진 공통부분과 잘 알려진 프로세스의 사용을 극대화하여 품질과 생산성을 향상시키며 유지보수에 능동적으로 대처할 수 있도록 계획되어진 체계화된 행위들의 집합이다.

소프트웨어 재사용에 있어서 기존의 라이브러리를 기반으로 한 재사용은 컴퍼넌트의 조합 과정이 쉬운 작업이 아니라는 점에서 또한 코드 재사용의 한계를 갖는다는 점에서 재사용의 효과를 극대화하지는 못하였다. 따라서 이러한 라이브러리와는 달리 한 영역에서의 재사용만을 지원하는 프레임워크 기반 애플리케이션에 초점을 둔다. 이는 한 도메인에 속한 다양한 애플리케이션 개발에 공통적으로 필요한 컴퍼넌트들과 컴퍼넌트들간의 상호 연결관계 및 상호 작용 방법까지도 포함하고 있기 때문에 한 영역에서의 기본 애플리케이션 행위를 정의하고 있다는 점이 라이브러리를 기반으로한 재사용과는 다르다.

컴퍼넌트기반의 소프트웨어는 객체 지향적인 소프트웨어 모듈이므로 재사용성을 높이고 과거에는 개발에 소요되는 비용의 증가에 비해 고급화되어 가는 고객의 욕구를 만족시킬 수 없었지만 컴퍼넌트는 재사용성으로 개발에 소요되는 비용을 상당히 줄일 수 있으며 적은 개발비로 사용자의 요구를 충족시킬 수 있다는 장점을 지니고 있다. 또한, 컴퍼넌트기반의 소프트웨어 경우 모듈화 되어 있으므로 기존의 소프트웨어처럼 유지보수에 막대한 비용을 들일 필요없이 문제가 되는 모듈 하나 하나를 유지보수해 나가면 되므로 유지보수 비용을 상당부분 절감할 수 있다는 강력한 대응책을 가지고 있다. 컴퍼넌트 소프트웨어 등장은 *DLL*(Dynamic Linking Library)이라는 기법이 도래된 후부터이다. 이는 *VBX*(Visual Basic controls with the file eXtension)라 불리는 커스텀 컨트롤 구조를 만들게 되는 계기가 되고 점점 확장되고 있다. 현재 실제적으로 쓰이고 있는 컴퍼넌트 소프트웨어들은 *OLE*(Object Linking Embedding), OpenDoc, Java, CyberDog등이 있다.

따라서 본 논문에서는 프레임워크 재사용을 위해 컴퍼넌트기반의 방법론을 제시하고 그 프레임워크 도메인 개발 환경을 구축한다. 즉, 컴퍼넌트 기반의 소프트웨어 생성을 위해 실세계의 도메인 지식을 입력으로 도메인을 분석하고 분석된 정보를 통해 hotspot을 식별하여 사용자와 개발자의 추가정보를 첨가하는 재설계 과정을 거친다. 이후 도메인에 따라 도메인 프레임워크와 애플리케이션 프레임워크를 생성한다. 이러한 컴퍼넌트 지향 방법론(COM : Component-Oriented Methodology)은 내부적으로는 컴퍼넌트/패턴 라이브러리 저장소를 통해 정보를 검색, 이해하여 추출하거나 혹은 합성으로 얻어진 정보는 각각 컴퍼넌트와 패턴에 대한 정보로 분류되고 이것은 재설계시 추가정보로 사용되어진다. 이로 인해 사용자는 멀티미디어 컴퍼넌트를 이용해 자신의 애플리케이션에 쉽게 적용 가능한 개발 환경 즉, 본 논문에서는 *CORBA*(Common Object Request Broker Architecture)[1]를 기반으로 한 컴퍼넌트/패턴 관리 시스템(Component Pattern Management System : CPMS)을 통해 멀티미디어 컴퍼넌트를 추가, 갱신, 삭제하는 기반 환경(infrastructure environment)을 구축한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 재사용성을 기반으로 한 프레임워크 도메인을 분류하고 *CBSE*(Component Based Software Engineering)의 기본적인 개념과 기존 방법론에 대해서 논한다. 그리고 제 3장에서는 컴퍼넌트기반의 프레임워크 도메인을 구축하기 위해 도메인 분석 및 모델링, 컴퍼넌트/패턴 뷰어(viewer), 컴퍼넌트 관리로 분류하여 설명한다. 또한, 제 4장에서는 실제로 프레임워크를 구성하기 위해 기존의 애플리케이션을 통해 분석과 설계를 한 후 사용자의 요구와 기존의 것을 고려하여 재설계 과정을 보인다. 또한 실제로 멀티미디어 컴퍼넌트인 광고 컬럼 애플리케이션(Advertisement Column Application)을 구성하고 전체 시스템 구조에 적용하기 위해 CORBA로 그 하부기반을 정립한다. 제 5장에서는 평가로 본 논문에서 제시한 컴퍼넌트/패턴 관리 시스템을 개발 방법론 측면과 재사용 시스템 측면을 고려하여 비교한다. 마지막으로 제 6장에서는 결론 및 향후 연구 방향을 제시한다.

2. 관련연구

2.1 프레임워크 재사용(Framework Reuse)

컴퍼넌트의 재사용은 사용자에게 제공되는 인터

페이스 환경의 열악함 그리고 유지보수를 위한 높은 비용 때문에 실패를 거듭하고 있다. 그러므로 한단계 높은 프레임워크 연구의 필요성이 반듯이 요구되며 재사용 가능한 부품을 분류하여 저장하거나 부품 검색 혹은 이해, 그리고 수정단계에 대한 연구도 매우 필요하다[2].

프레임워크는 추상화된 클래스들의 집합 또는 특정 도메인에서 재사용 가능한 설계를 구성하는 상호 연관된 클래스들의 집합이다. 프레임워크를 기반으로 한 애플리케이션의 개발은 기존의 개발방법과는 차이점을 가진다. 즉, 기존 애플리케이션 개발 환경에서의 재사용이란 함수와 클래스들의 집합체인 라이브러리를 제공하고 개발자들은 라이브러리에서 필요한 컴퍼넌트들을 식별한 후 이들 컴퍼넌트들을 조합함으로써 새로운 애플리케이션을 개발한다. 라이브러리를 통한 재사용에서 가장 중요한 요소는 올바른 컴퍼넌트를 식별하고 컴퍼넌트의 조합 방법을 정확하게 파악하는 것이므로 라이브러리의 분류 기법 및 검색 시스템의 정확성이 재사용의 성공을 결정하는 요인이다. 그러나 이러한 라이브러리를 기반으로 한 재사용은 컴퍼넌트의 조합 과정이 쉬운 작업이 아니라는 점에서 또한 코드 재사용의 한계를 갖는다는 점에서 재사용의 효과를 극대화하지는 못하였다. 이를 극복할 수 있는 방법으로 프레임워크 기반 애플리케이션을 개발한다. 이는 라이브러리와 달리 한 영역에서의 재사용만을 지원한다. 한 도메인에 속한 다양한 애플리케이션 개발에 공통적으로 필요한 컴퍼넌트들과 컴퍼넌트들간의 상호 연결관계 및 상호 작용 방법까지도 포함하고 있기 때문에 한 영역에서의 기본 애플리케이션 행위를 정의하고 있다는 점이 라이브러리를 기반으로한 재사용과는 다르다. 그러므로 프레임워크를 재사용하여 애플리케이션을 개발하는 것은 프레임워크가 이미 애플리케이션의 행위를 포함하고 있으므로 컴퍼넌트들의 조합을 정의하는 것이 아니라 애플리케이션의 기본 행위를 구성하게될 컴퍼넌트를 제정의하는 것이다. 프레임워크 자체를 개발하는 것은 애플리케이션을 개발하는 것보다 더 많은 노력과 검증이 필요하다.

이러한 프레임워크는 애플리케이션을 개발할 수 있는 아키텍쳐를 제공하는 면에서 라이브러와는 차이가 있다. 프레임워크 설계와 코드 부분을 재사용함으로 개발비용과 시간이 감소되며 사용이 용이하지만 구현이 힘들다. 일반적으로 프레임워크 도메인은 <표 1>과 같이 내부 구조에 의한 분류와 문제 영역에 의한 분류로 나눈다[3].

표 1. 프레임워크 도메인 분류
Table 1. Framework Domain Catagory

내부구조에 의한 분류	문제 영역에 의한 분류
블랙 박스	응용 프레임워크
화이트 박스	도메인 프레임워크
그레이 박스	지원 프레임워크

2.2 CBSE(Component-Based Software Engineering)

컴퍼넌트는 넓게는 접근 가능한 인터페이스를 제공하는 모듈을 나타내고 좁게는 일반적으로 널리 알려진 CORBA, *COM*(Component Object Model), JavaBeans등을 의미한다[4]. 컴퍼넌트라는 개념은 뚜렷이 명시되어있지는 않으며, 최근 *VB*(Visual Basic), Delphi등이 나오면서 많이 거론된 용어로 다음 <표 2>와 같이 여러 정의를 내리고 있다[5,6,7].

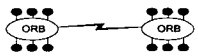
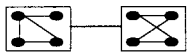
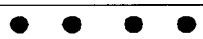
표 2. 컴퍼넌트 정의
Table 2. Component Definition

연 구 자	정 의
Philippe Krutchen	잘정의된 아키텍쳐상에서 어떠한 기능을 수행하는 시스템의 독립적이면서 대치가능한 부분이다. 컴퍼넌트는 인터페이스들의 집합에 대한 물리적인 구현을 제공한다.
Gatner Gruop	동적으로 바인드할 수 있는 하나 이상의 프로그램들을 하나의 단위로 관리하는 패케지로서 실행시간에 발견될 수 있는 인터페이스를 통해 접근이 가능하다.
Szyperski	계약상으로 명시된 인터페이스들과 명확한 문맥 의존성을 가진 컴포지션의 단위이다.
Kozaczynski	자발적인 비즈니스 객체 또는 비즈니스 로직을 소프트웨어로 구현한 것이다.

소프트웨어 공학이 대두된 1970년이래, 대부분의 사람들이 컴퍼넌트와 컴퍼넌트기반 설계에 관한 것을 논의하면서 CBSE의 개념을 정의하였다. 즉, 소프

트웨어 공학은 컴퍼넌트기반을 위한 기술과 툴을 완벽하게 지원하지 못함으로 CBSE 즉, 컴퍼넌트웨어와 *CBD*(Component Based Development)라 불리는 새로운 개념이 등장하였다. 일반적으로 이러한 컴퍼넌트와 연관된 전체적인 패턴의 범주는 <표 3>과 같이 분류할 수 있다.

표 3. 컴퍼넌트 관련 패턴
Table 3. Component Patterns

종 류	도식화
기업형 구조	
시스템 구조	
응용 구조	
매크로 구조	
마이크로 구조	
객체	

즉, 단순한 객체의 개념에서 상위로 갈수록 그 객체간의 관계성, 또한 일 대 일(1:1)의 관계가 아니라 다 대 다(m:m)의 관계성을 가진 형태, 그 자체의 모듈화, 모듈화한 컴퍼넌트간의 관계성, 그리고 하부기반을 가진 컴퍼넌트와 그 인터페이스를 정의할 수 있다.

이러한 CBSE는 단순히 컴퍼넌트를 개발하거나 얻는 것뿐만이 아니라 소프트웨어 집약적인 시스템의 생산과 개발로 조립식 컴퍼넌트를 유도하는데 있다. 그 주요 구성요소로 컴퍼넌트, 인터페이스(interface)를 들 수 있는데 컴퍼넌트는 잘 정의된 아키텍쳐 환경에서 명확한 기능을 이행하는 특별한 그리고 독립적이며, 시스템의 대치 가능한 부분 혹은 인터페이스 집합의 물리적 실현을 따르고 제공하는 것이라고 할 수 있다. 또한, 이것은 크게 *OOP*(Object-Oriented Programming), *OOA*(Object-Oriented Analysis) & *OOD*(Object-Oriented Design), 시각화 프로그래밍, 하부기반, 기업 애플리케이션으로 분류하여 정의할 수도 있다. 인터페이스는 컴퍼넌트를 서비스하는데 기술하는 오퍼레이션의 집합이다[8,9].

2.3 기존의 방법론

컴퍼넌트기반 소프트웨어 개발의 모델은 전형적인 개발 모델 즉, 객체지향 방법론(OOM : Object-Oriented Methodology)과 다르다. <표 4>는 전형적인 소프트웨어 개발 방법론과 컴퍼넌트기반 소프트웨어 개발의 주요 특징을 요약하였다[10]. 또한, 특징 각각에 대해서 살펴보았다.

표 4. 개발 모델의 비교
Table 4. Comparison of Development Model

특징	OOSE	CBSE
아키텍쳐	단일체	모듈
컴퍼넌트	구현 & 화이트박스	인터페이스 & 블랙박스
프로세스	빅뱅 & 폭포수형	전개 & 동시
방법론	스크래취로부터 개발	합성
조직체	단일체	전무화 : 컴퍼넌트 벤더, 브로커, & 통합자

- **아키텍쳐**

CBSE는 모듈 아키텍쳐를 강조하는데 이러한 설계를 만들기 위해 소프트웨어 시스템 즉, 소프트웨어 아키텍쳐의 확실한 기반을 필요로 한다. 또한, 대부분 컴퍼넌트기반 시스템은 *MFC*(Microsoft Foundation Class)와 CORBA같은 소프트웨어 아키텍쳐에 기초를 두며 프레임워크의 형태로 제공한다.

- **컴퍼넌트**

컴퍼넌트는 특별한 생산물, 특별한 도메인 혹은 도메인에 독립된 것을 의미한다.

- **프로세스**

CBSE는 전개적(Evolutional)으로 만들어지며 시스템의 이러한 부분이 컴퍼넌트 벤더 또는 다른 조직체에서 조립하므로 소프트웨어 프로세스의 몇몇은 동시에 진행한다.

- **방법론**

방법론들은 컴퍼넌트 개발과 컴퍼넌트 합성 모두를 요구하며 객체지향 방법론과 같은 전형적인 방법론들의 대부분은 스크래취로부터 개발하고 재사용 기반 개발에 큰 효과를 얻지는 못했다. 그러나, 소프트웨어 컴퍼넌트는 구현으로부터 인터페이스를 분리하고 인터페이스를 제공한다.

• **조직체**

컴퍼넌트 개발과 컴퍼넌트 통합의 분리는 컴퍼넌트 브로커의 새로운 역할을 생성하며 컴퍼넌트 브로커는 소프트웨어 컴퍼넌트를 팔거나 분산시키며 컴퍼넌트 개발과 컴퍼넌트 통합이 서로 다른 전문적인 기술을 제공함으로 컴퍼넌트 벤더와 컴퍼넌트 통합자에게 그대로의 조직체를 특수화한다.

3. 컴퍼넌트기반 방법론 (Component-Oriented Methodology)

컴퍼넌트 기반의 소프트웨어 생성을 위해 실세계의 도메인 지식을 입력으로 도메인을 분석하고 분석된 정보를 통해 hotspot을 식별하여 사용자와 개발자의 추가정보를 첨가하는 재설계 과정을 거친다. 이후 도메인에 따라 도메인 프레임워크와 애플리케이션 프레임워크를 생성한다. 이는 내부적으로 컴퍼넌트/패턴 라이브러리 저장소를 통해 정보를 검색, 이해하여 추출하거나 혹은 합성으로 얻어진 정보는 각각 컴퍼넌트와 패턴에 대한 정보로 분류되고 이것은 재설계시 추가정보로 사용되어진다. 또한, 컴퍼넌트/패턴 관리 시스템을 통해 컴퍼넌트/패턴을 검색, 갱신, 삭제가 가능하다. 다음 (그림 1)은 전체 시스템 구성도를 나타낸다. 즉, 도메인 분석 및 모델링, 컴퍼넌트/패턴 뷰어, 컴퍼넌트/패턴 관리로 나누어 정의한다.

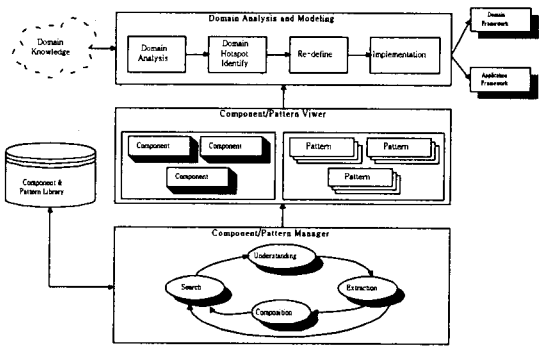


그림 1. 시스템 구조도
Figure 1. System Architecture

3.1 도메인 분석과 모델링

일반적으로 프레임워크 개발을 시작하는 단계에서 우선적으로 결정해야할 일은 도메인을 정의하는 것이다. 단일 프레임워크로 일반적인 문제점을 모두 해결하기는 불가능하므로 도메인을 정의하는 것은 중요하다. 프레임워크 개발에 있어서 결정된 도메인에 따라 요구사항이 그 도메인에 있는지 아닌지를 판단할 수 있다. 또한, 프레임워크가 필요한 애플리케이션에 대해 재사용을 할 수 있는지 없는지를 결정할때도 도메인을 정하는 것이 필요하다.

이러한 도메인이 정해지면 이 도메인에 대한 분석이 이루어진다. 이때 여러 애플리케이션에 대해 일반적인 요구사항과 각 애플리케이션에 특정한 요구사항 중 어디에 초점을 맞추느냐는 무엇보다 중요하다. 요구사항을 얻는다는 것은 애플리케이션들 사이에 일반적인 모든 요구사항들을 고립시키는 것과 이러한 요구사항들을 프레임워크에 부과된 요구사항으로 만드는 것이다.

이러한 분석된 정보는 공통된 기능을 추출하는데 있어서 일반적으로 이용 사례도를 사용하게 되는데 이때 객체와 그들간의 협동관계가 정의되고 클래스와 그들의 메소드를 자세히 기술한다. 프레임워크를 설계하는 동안 여러 가지 유사한 애플리케이션을 적용하게 되는데 이것은 구현에 대한 기초를 제공해 주기도 한다.

3.2 컴퍼넌트/패턴 뷰어

식별된 정보는 사용자가 요구하는 것에 대한 해결책을 찾기 위해 유사한 정보를 보기(viewing) 원하며 이를 위해 (그림 2)과 같이 컴퍼넌트/패턴을 사용한 것을 도식화한다. 즉, 컴퍼넌트/패턴 라이브러리에서 추출된 컴퍼넌트와 패턴들은 각각 개발 프로세스에 조립되거나 적용되어져 실세계 도메인 지식과 함께 새로운 정보를 구축할 수 있는 기반 지식이 된다. 컴퍼넌트로부터 시스템을 구축하기 위하여 서로 다른 유형의 서비스들을 제공하는 광범위한 멀티미디어 컴퍼넌트로부터 선택하는 것이 가능해야만 한다. 몇몇 경우에, 제공되는 행위는 많은 애플리케이션 도메인들에 공통된 기본적인 하부 서비스들을 관리하기 위한 것일 수 있다. 더 높은 추상화 수준에서 멀티미디어 컴퍼넌트는 또한, 특별한 도메인에 특정적인 많은 기능을 제공할 수 있다.

애플리케이션을 생성하기 위하여 컴퍼넌트를 조립

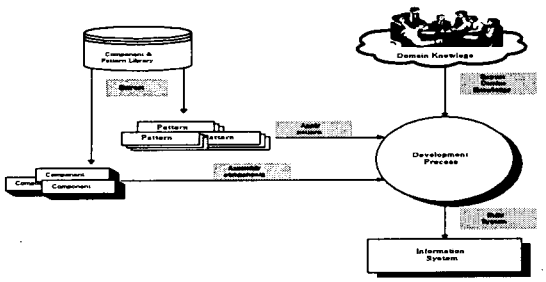


그림 2. 컴퍼넌트/패턴의 사용
Figure 2. Use of Components/Patterns

하는 것은 멀티미디어 컴퍼넌트에 의해서 제공되는 서비스에 대한 세부사항이 잘 묘사되고 견고하고 완벽할 것을 요구한다. 설계기간 동안 중심이 되고 구현기간 동안 멀티미디어 컴퍼넌트 사용자와 벤더(vender)사이에 중개자를 제공하는 것은 인터페이스가 담당한다. 대규모의 기업에서 애플리케이션을 구축하기 위해서는 컴퍼넌트 인터페이스를 기술하기 위한 풍부한 언어가 모든 컴퍼넌트기반 개발 접근방법의 핵심적인 측면이다.

또한, 유사성을 구축하기 위해 패턴은 부분들을 결합하기 위한 규칙이다. 청사진을 그릴 때 이러한 패턴들이 컨텍스트(context)로 알려진 것에 적용된다. 이론적으로 잘 정의된 패턴은 단순히 제공된 컨텍스트 파라미터들을 가질 수 있고 대부분의 청사진은 자동적으로 그려지고 따라서 개발자는 상당한 양의 노력을 줄일 수 있다[11].

3.3 컴퍼넌트/패턴 관리

사용자는 원하는 멀티미디어 컴퍼넌트의 검색(searching)을 위해서 제시되어진 패싯 항목을 선택함으로써 가능한데 패싯 검색은 시스템에서 제공되는 여러 가지 패싯들 중 찾으려 하는 멀티미디어 컴퍼넌트에 알맞은 속성을 나타내는 항목들을 선택하여 합성함으로써 특정 멀티미디어 컴퍼넌트를 검색한다.

원하는 멀티미디어 컴퍼넌트를 검색한 후 자신의 프로그램에 통합이 가능한지를 결정하는데 도움을 주기 위해 사용자의 이해 정도에(understanding) 따라 멀티미디어 컴퍼넌트 정보를 즉각적이고 구체적인 시각적 형태의 정보를 제공하는데 그 형태는 일반적으로 파일, 하이퍼링크, 텍스트 그리고 그래픽등이 있다.

이러한 정보를 통해 사용자가 원하는 멀티미디어 컴퍼넌트를 추출(extraction)한다. 추출 시, 두 가지 패싯 항목인 운영체제(OS : Operating System)와 언어(Language) 모두를 만족하는 것을 추출한다. 이렇게 획득되어진 재사용 멀티미디어 컴퍼넌트들은 사용자가 원하는 요구를 충분히 만족시킬 수 있도록 합성(composition)을 통해 얻을 수 있는데 이것은 원시코드 수준의 클래스간의 합성이 아닌 실행가능한 멀티미디어 컴퍼넌트를 서로 다른 언어, 운영체제등에 독립적으로 사용할 수 있다. 일반적인 애플리케이션에서 기본적으로 제공하지 않는 기능, 즉, 다양한 멀티미디어의 데이터 표현과 사용자와 상호 작용하는 기능을 제공해주는 소프트웨어 모듈인 Plug & Play방식등이 있다.

4. 사례연구

본 논문에서는 컴퍼넌트기반의 소프트웨어 생성을 위해 기존의 애플리케이션을 분석하고 설계한 후 사용자의 반복되는 요구사항을 hotspot으로 인식하여 기존의 분석결과를 기반으로 추가할 분석사항을 작성하는 재설계 과정을 거침으로 사용자와 개발자의 요구사항을 적절히 수용하는 새로운 멀티미디어 컴퍼넌트를 구성한다.

4.1 도메인 분석 및 모델링

멀티미디어 컴퍼넌트 프레임워크 구축을 위해 본 논문에서는 블랙박스 형태의 프레임워크를 구성한다. 애플리케이션 개발자는 클래스를 인스탄스화하고 그들의 멤버함수를 호출함으로써 프레임워크의 built-in 기능을 사용하고(black-box), 새로운 클래스를 파생시키고 멤버함수를 오버라이드(override)함으로써 기능을 확장 및 수정(white-box)하게 되는 것이다. 또한, 프레임워크를 사용하여 애플리케이션들이 개발되어짐에 따라, 선택할 수 있는 서브클래스들의 라이브러리가 만들어진다.

4.1.1 도메인 지식을 통한 도메인 분석

기존의 웹상에 부유상태의 광고 컬럼 애플리케이션 중 하나를 선택하여 분석하였다. 이것은 다음과 같은 (그림 3)처럼 문제 명세서를 기반으로 한다.

광고 컬럼 애플리케이션은 사용자가 여러 종류의 상품중 원하는 상품을 선택하게 되면 해당상품을 서버에서 검색한 후 요약정보와 그 목록이 제시된다. 제시된 정보보다 좀더 상세한 내용을 원할 경우 링크된 부분을 클릭하면된다. 제시된 상세정보는 상품의 이미지와 상품의 제원에 대해서 나타나며 구매를 원할 경우 색상이나 개수, 지불방법등에 대한 구매서를 작성한 후 주문을 하게된다.

그림 3. 문제 명세서
Figure 3. Problem Specification

영역분석을 통해 사용 사례도를 (그림 4)와 같이 *UML* (Unified Modeling Language) 표기법을 사용하여 나타내었으며 시스템 내의 객체 상호작용을 모형화하기 위해(그림 5, 6)과 같이 수직선상의 여러개의 오브젝트와 순서나 함수로 시간이 지나가는 것에 따라 오브젝트들 사이에 메시지 교환을 나타내는 Sequence 다이어그램과 객체와 그들만의 관계를 동적인 협력사항으로 보여주는 Collaboration 다이어그램으로 표현하였다.

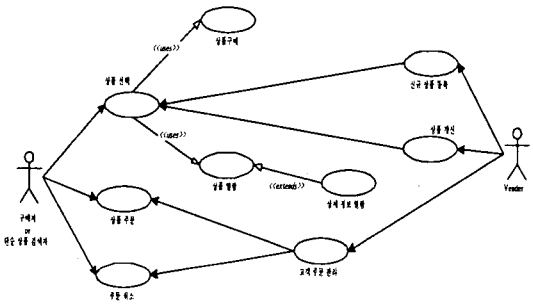


그림 4. 사용사례도
Figure 4. Use Case

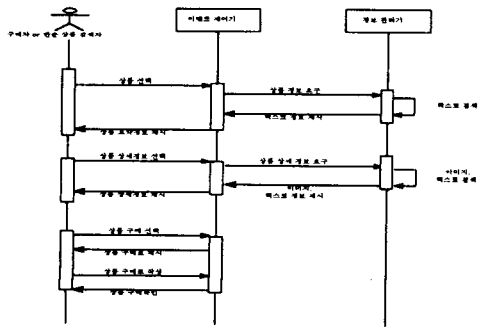


그림 5. sequence 다이어그램
Figure 5. Sequence Diagram

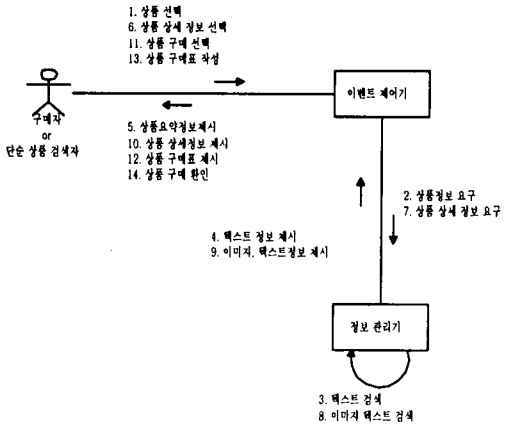


그림 6. Collaboration 다이어그램
Figure 6. Collaboration Diagram

또한, 추출된 공통적인 클래스와 객체들은 (그림 7)과 같이 시스템에서 다루어지는 클래스들의 정적인 구조를 연관, 종속, 특수화된 또는 패케이지화된 여러 가지 방법으로 다른것들과 각각 관계를 가지는 것을 표현한 Class 다이어그램으로 나타내었다.

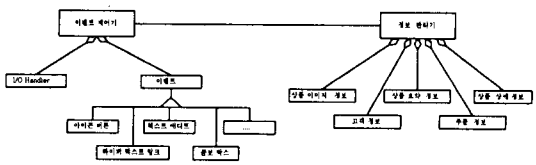


그림 7. Class 다이어그램
Figure 7. Class Diagram

4.1.2 Hotspot 식별

Hotspot의 식별은 블랙박스 프레임워크에서 매우 중요하며 애플리케이션마다 있을 수 있는 다양한 변화의 부분을 hotspot으로 정의할 수 있다. 이것은 재정의된 메소드에 대응되며 융통성과 일반성을 지니며 잘 설계된 프레임워크 개발에 도움을 준다.

앞 절에서 영역분석을 통해 클래스를 추출하였고 공통적으로 수행하는 부분과 추출된 객체들에 위치하는 반복해서 쓰는 유사한 코드를 식별한다. 즉, 텍스트 항목, 이미지, 오디오, 동영상 등을 hoptspot으로 간주하고 (그림 8)는 본 논문에서는 텍스트와 이미지만을 고려한 것을 제시한다. 이러한 정보는 다음 절에 제시 할 재설계에서 사용자 측면에서 분석된 정보 즉 hotspot와 개발자가 요구하는 정보를 충족하

게끔 한다.

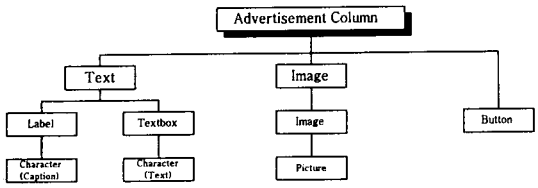


그림 8. Hotspot 식별
Figure 8. Hotspot Identify

4.1.3 리팩토링(Refactoring)

개발중의 재설계를 의미하는 것으로 리팩토링(refactoring)은 기능 추가를 동시에 하지 않으며 가능한 많은 테스트를 하고 작업은 짧고 신중하게 해야 하는 것을 원칙으로 하고 있다. 개발 초기에는 소프트웨어 전체 기능을 작게 잡아 시작하다가 개발중에 기능이 점차 추가되어 처음에 생각했던 소프트웨어보다 훨씬 많은 기능을 수행하게되는데 이때 "소프트웨어 엔트로피[12]" 즉, 개발초기에는 문제가 없었던 설계가 여러 가지 기능을 첨가함으로 엉망이 되는 현상이 발생된다. 이는 추가된 기능을 반영해 설계를 바꾸지 않고 기존 설계위에 추가된 내용을 덧붙이기 때문이다. 이는 재설계를 함으로 가능하기는 하나 재설계에 소모되는 노력과 시간도 문제이지만 기존의 프로그램을 고치면 새로운 버그나 문제점이 등장할 확률이 많다. 따라서 위에서 언급된 원칙을 통해 개발중에 재설계를 의미하는 리팩토링을 한다. 새로운 기능을 추가하려고 하는데 기존의 코드 때문에 불가능한 경우나 자신이 작성한 설계나 코드를 이해하기 힘든 경우에는 리팩토링을 해야한다.

따라서 본 논문에서는 기존의 애플리케이션 분석을 통해 제시된 것에서 hotspot으로 식별되어진 것을 인식하고 몇몇 기능을 추가하고자 리팩토링을 한다. 다음 (그림 9, 10, 11, 12, 13)은 각각 사용자의 새로운 요구사항을 충족시킨 문제 명세서, 사용 사례도, sequence 다이어그램, collaboration 다이어그램, class 다이어그램이다.

추가적으로 UML 지식 기반으로 동적이고 다중적인 경로를 통해 네비게이션 되는 정보와 구성요소에 초점을 주고 웹 애플리케이션에 대한 분석과 설계를 위한 네비게이션 다이어그램을 표현한다. 우선 Simple

광고 컬럼 애플리케이션 프레임워크는 하나의 입력창을 이용하여 광고 내용을 담은 HTML문서 생성작업을 한다. 사용자가 만들 광고에 대한 기본항목을 제시된 폼에 입력하고 상품 이미지를 선택한다. 입력이 끝나면 HTML 문서 생성 이벤트에 의해 광고란 프레임워크는 사용자 웹페이지에 삽입할 수 있는 광고 웹 문서를 얻을 수 있다.

그림 9. 문제 명세서
Figure 9. Problem Specification

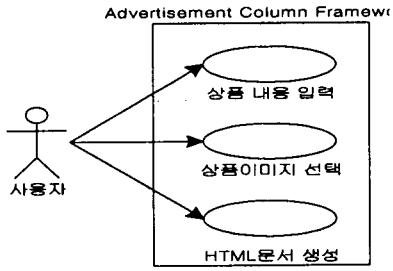


그림 10. 사용 사례도
Figure 10. Use Case

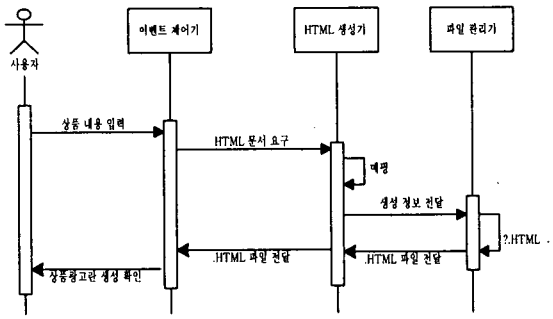


그림 11. Sequence 다이어그램
Figure 11. Sequence Diagram

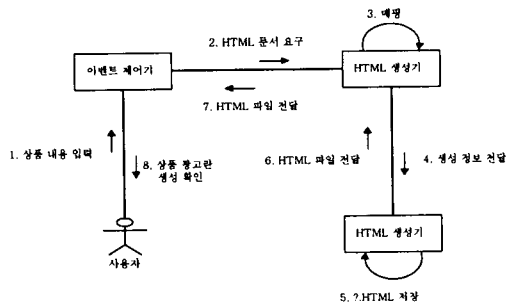


그림 12. Collaboration 다이어그램
Figure 12. Collaboration Diagram

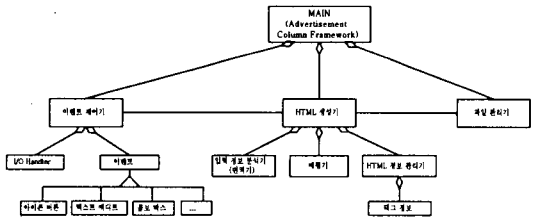


그림 13. Class 다이어그램
Figure 13. Class Diagram

네비게이션은 인터넷/인트라넷뿐만 아니라 클라이언트/서버 환경에서 네비게이션 되는 정보의 흐름과 연결에 관련된 전반적인 사항을 나타내기 위한 것이다. 전송되는 자료는 시스템 이벤트와 사용자 선택에 의해 정해지며 이때 메시지는 웹 구성요소사이에서 네비게이션 된다.

또한, Use 네비게이션은 전체 시스템에서 사용되는 오퍼레이션 구성요소의 사용을 나타내며 네비게이션 되는 동안 전달되는 자료의 목적지를 명시하는 것을 원칙으로 한다. 이는 사용자에게 시스템의 전반적인 구조와 실매개체를 직관적으로 파악하고 이해할 수 있는 장점을 가진다. 다음 (그림 14)은 사용 네비게이션의 표기법을 나타내며 네비게이션 다이어그램에 연결관계를 지원한다. <표 5>는 그 기호에 대한 설명이다.

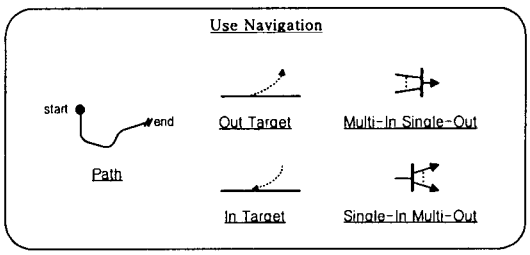


그림 14. 사용 네비게이션 다이어그램 기호
그림 14. Use Navigation Diagram Notation

다음 (그림 15)는 위의 기호를 이용해 본 논문에서 적용된 네비게이션 다이어그램이다[13].

4.1.4 프레임워크 구축

본 논문에서는 분석과 설계를 통해 정의된 소프트웨어를 하나의 모듈화로 정의하기 위해 앞 절에서 정의된 분석과 설계를 통해 직접 멀티미디어 컴퍼넌트

표 5. 사용 네비게이션 다이어그램 기호명과 내용
Table 5. Use Navigation Diagram Notation Name and Contents

Path	구성요소를 네비게이션하는 모든 경로를 표현
Out Target	네비게이션되는 도중 정보가 도달되는 목적지와 정보를 표현
In Target	네비게이션되는 도중 유입될 수 있는 정보와 그 위치를 표현
MISO (Multi-In Single-Out)	다중적인 경로를 표현하기 위한 것으로 하나 이상의 경로가 단일의 경로로 합쳐지는 것으로 구성요소에서 유도될 수 있음을 표현
SIMO (Single-In Multi-Out)	MISO와 유사하게 단일의 구성요소에서 다중적인 경로로 뻗어나는 것을 표현

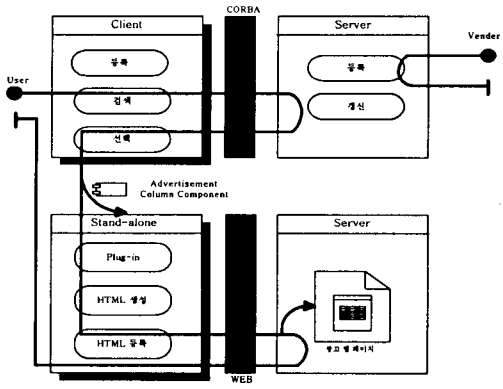


그림 15. 시스템 네비게이션 다이어그램
Figure 15. System Navigation Diagram

프레임워크를 구성한다.

구성할 광고 컬럼 애플리케이션 프레임워크는 영역 프레임워크(domain framework)이고, 이러한 프레임워크는 각 애플리케이션 영역마다 다양하게 개발 가능한 도메인이지만 텍스트 항목, 이미지, 오디오, 동영상등을 지원하는 환경은 모두 유사하기 때문에 공통적인 아키텍쳐를 정의할 수 있다. 또한, 분야마다의 특징에 따라 사용자가 임의로 변경할 수 있는 hotspot 역시 미리 추출이 가능하고 이를 캡슐화하거나 정련화된 알갱이의 객체를 만들므로 블랙박스 프레임워크의 재사용 부품인 pluggable한 객체 개발이 가능하다.

다음(그림 16)은 광고 컬럼 애플리케이션에 대한 전체 구성도이며 stand-alone환경에서 사용자의 입력에 대한 디자인 폼을 제시하며 html문서가 자동 생성되는 것을 볼 수 있다. 또한, 사용자가 쉽게 웹상에 멀티미디어 컴퍼넌트를 올려서 사용하는 것을 보여준다. 본 논문에서는 VB 5.0에서 ActiveX 컨트롤을 사용하여 생성하였다.

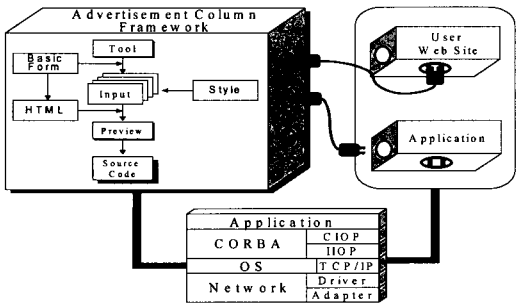


그림 16. 광고 컬럼 애플리케이션 구성도
Figure 16. Advertisement Column Application Construction Diagram

4.4 컴퍼넌트/패턴 관리

사용자의 요구에 맞는 멀티미디어 컴퍼넌트를 검색, 추출하여 pluggable함으로 사용자는 쉽게 그 멀티미디어 컴퍼넌트를 사용할 수 있다. 사용자는 컴퍼넌트 관리 시스템을 통해 멀티미디어 컴퍼넌트를 관리하며, 서버에 있는 멀티미디어 컴퍼넌트를 선택하여 사용자 환경에서 stand-alone으로 그 멀티미디어 컴퍼넌트를 쉽게 사용할 수도 있고, 웹상에 올려서 사용할 수도 있다. 본 논문에서는 앞 절에서 구성한 광고 컬럼 애플리케이션인 멀티디미어 컴퍼넌트를 선택하여 독자적으로 혹은 웹상에 올려지는 것을 제시한다.

본 논문에서 구현은 클라이언트측은 Win 95상에서 CORBA상에서 운용되며 컴퍼넌트 관리 시스템은 다음 (그림 17, 18)과 같이 멀티미디어 컴퍼넌트를 검색하거나 사용자가 프레임워크를 통해 생성한 멀티미디어 컴퍼넌트를 등록할 수도 있다. 컴퍼넌트 관리 시스템은 VC++ 5.0[14]을 사용하여 구현하였으며 서버측은 Win95상에 CORBA상에서 운영된다.

다음 (그림 18)에서 컴퍼넌트 분류는 일반적으로 멀티미디어 컴퍼넌트들의 공통된 특성을 수집하여 패싯을 구성한 후 멀티미디어 컴퍼넌트에 알맞은 속성

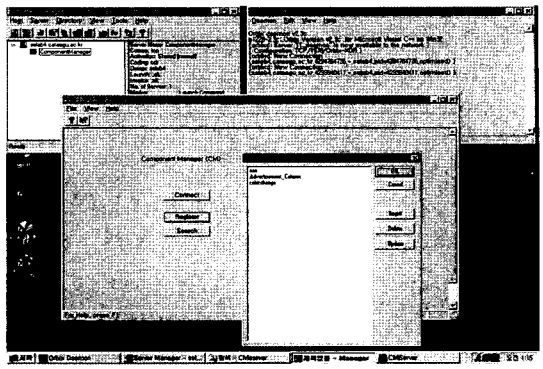


그림 17. 컴퍼넌트 관리 시스템 - 컴퍼넌트 등록
Figure 17. Component Management System-Component Register

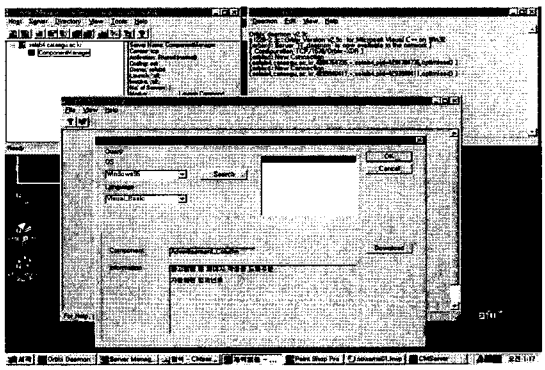


그림 18. 컴퍼넌트 관리 시스템 - 컴퍼넌트 검색
Figure 18. Component Management System-Component Search

을 패싯항목으로 선택하고 조립하는 과정을 사용한다. 각 패싯은 멀티미디어 컴퍼넌트들의 대표 특성들을 나타내는 항목들로 이루어지므로 새로운 멀티미디어 컴퍼넌트 추가에 융통성을 제공하여 항목이 지속적으로 증가할 때 신축성있게 대처할 수 있다. 또한, 멀티미디어 컴퍼넌트에 대한 효과적인 표현이 가능해 정확한 멀티미디어 컴퍼넌트의 검색이 용이하다. 멀티미디어 컴퍼넌트의 공통적인 속성을 모아 하나의 패싯으로 모으고 여러 가지 패싯들 중 찾으려하는 멀티미디어 컴퍼넌트에 알맞은 속성을 나타내는 항목들을 선택하여 합성함으로써 특정 멀티미디어 컴퍼넌트를 검색할 수 있다.

컴퍼넌트 관리에 있어서 현재, 이 기종간의 클라이언트/서버 시스템을 더욱 효율적으로 통합, 관리할 수 있는 시스템을 요구하게 되었고 여기에 부합해

객체지향 기술을 이용한 관련 표준을 제정하기 위해 OMG가 설립되었으며 그 표준중 가장 중요한 부분이 바로 CORBA라는 미들웨어이다.

이를 지원하는 제품중 Orbix 2.3c는 멀티스레드를 지원함으로써 성능이 뛰어나며 대부분의 기업들이 이 제품을 이용하고 있다. 클라이언트로부터의 요청과 답변에 대한 필터링 기능을 제공함으로써 개발자가 독자적으로 압축/해독 및 암호화/복원화 기능을 추가할 수 있다.

본 논문에서는 VC++ 5.0으로 Orbix 2.3c를 사용하여 미들웨어를 구성하는데 일반적으로 CORBA 응용 프로그램 개발 과정은 다음과 같다[15,16].

① IDL 작성

인터페이스 부분을 정의하는 언어로 다음 <리스트 1>에서 CompMgr.idl을 작성한다.

리스트 1. IDL(CompMgr.idl) 정의
List 1. IDL Definition

```
ifndef i_CompMgr_IDL
#define i_CompMgr_IDL

#include "CommonDataType.idl";

interface CompMgr{
  oneway void start();
  void getComponentInfoList(out ComponentInfoList
clist);
  void searchComponentList(in Facet facet, out
ComponentInfoList clist);
  void registComponent(in ComponentInfo info,
in File file);
  void removeComponent(in ComponentInfo info);
  void updateComponent(in ComponentInfo info,
in File file);
  void downloadComponent(in ComponentInfo info,
out File file);
};

#endif
```

② IDL Interface 컴파일

위에서 작성된 인터페이스 CompMgr.idl화일은 컴파일 과정을 거치면서 C++ 매핑 규칙에 따라 C++ 파일로 변환되어진다. 이때 네트워크의 서로 다른 곳이나 서로 다른 메모리 공간내에 위치된 클라이언트와 서버간의 통신 기능 묘사를 위해 CompMgr.hh, CompMgrS.cpp, CompMgrC.cpp, CompMgr.ih, CompMgr.ic화일이 자동으로 생성되며 CompMgr.ih를 CompMgr_i.h로 CompMgr.ic는 CompMgr_i. cpp로 바꾼후 서버를 구현한다. 분산 객체에서 클라이언트 부분은 서버에 정의된 클래스의 메소드를 원격지에서 호출하고 이 결과를 전달받는 과정을 통해 작업한다. 그러므로 서버에서는 요구에 대한 메소드를 구현해야 한다. 이는 일반적인 C++ 프로그램 작성에서와 같이 인터페이스 설명을 위한 헤드 부분과 그 메소드의 실제 구현하는 두 개의 부분으로 이루어진다.

③ Server와 Client main 작성

클라이언트 부분과 서버부분에 필요한 모든 클래스 작성이 완료한 후 각 부분에 필요한 2개의 메인 프로그램을 작성한다. 클라이언트의 메인 프로그램은 해당 클래스를 선언하고 서버의 구현 부분과 해당 클래스를 바인딩 한 후 필요한 클래스를 호출하는 것이며, 서버의 메인 프로그램은 클라이언트가 요구 즉시 사용할 수 있도록 환경을 설정하고 메소드의 구현 부분을 완성하는 부분이다.

검색된 결과를 컴퍼넌트 관리 시스템의 인터페이스를 반환하게 되는데 이때는 작성된 인터페이스에 맞는 드라이버를 구동한다. 또한 컴퍼넌트 관리 시스템을 실행시키며 멀티미디어 컴퍼넌트를 추가, 삭제, 등록할 수 있다.

5. 평 가

본 논문에서는 개발 방법론 측면과 재사용 시스템 두 측면을 고려하는데 기존의 개발방법론들과 구축된 시스템 자체의 특성을 개발자가 제시한 기준으로 유사한 시스템과의 상대적인 비교를 통하여 구축 시스템이 가진 독창적인 성격을 중심으로 평가한다.

우선, 개발 방법론 측면에서 살펴보면, 기존의 객체지향 방법론을 기본으로 하는 Rose의 경우는 그 자체로 모든 작업을 수행한다. 즉, 컴퍼넌트 인터페이스 설계와 컴퍼넌트 구현 설계가 모두 객체지향 모델링 도구인 Rose안에서 수행되므로 작업을 일관되게 반복적으로 수행할 수 있지만 인터페이스 설계가 구현 설계에 의하여 지나치게 변경될 여지가 많다. 또한, Sterling의 Cool Family의 경우는 작업이

단계별로 목표를 갖고 분할되어 수행할 수 있지만 전체과정이 폭포수형으로 수행될 수 있다는 단점이 있다.

본 논문에서 제시한 방법론은 도메인 분석과 모델링 과정에서 실세계의 도메인 지식을 입력으로 도메인을 분석하는데 이때, UML을 기반으로 하는 인터넷/인트라넷에서의 네비게이션되는 정보의 흐름 순서는 시스템 이벤트에 의해 정해진다. 이 경우의 네비게이션되는 메시지가 장치사이의 흐름을 명시하기 위한 것으로 네비게이션 다이어그램을 제시하였다. 분석된 자료를 통해 hotspot을 식별하여 애플리케이션마다 있을 수 있는 다양한 변화의 부분을 정의할 수 있다. 이것은 재 정의된 메소드에 대응되며 융통성과 일반성을 지니며 잘 설계된 프레임워크 개발에 도움을 준다.

두 번째로 재사용 시스템 측면에서의 평가는 기존의 본 연구실에서 제시한 MT_View의 경우는 클래스 단위로 분류 모델을 다양하게 열거와 패싯을 정의하고 있고 검색 모델 역시 질의, 인덱스, 브러우징으로 검색을 한다. 또한, 사용자에게 이해정보를 제시한다. 하지만 다양한 모델 지원으로 인한 사용자의 혼란함을 초래하며 너무 서버에 의존한다는 단점이 있다. 따라서, 본 논문에서 제시한 컴퍼넌트/패턴 관리시스템의 경우 멀티미디어 컴퍼넌트 분류는 패싯 방법으로 했으며 사용자 인터페이스는 사용자의 직관적인 개념으로 쉽게 사용 가능할 수 있도록 했으며 복잡한 구성보다는 간단하고 단순한 구성으로 작업의 효율성을 높이려했다. 또한, 이해정보를 통해 사용자에게 편이성을 제공하고 멀티미디어 컴퍼넌트 관리에 있어서 등록, 삭제, 갱신을 주축으로 전개하였다.

6. 결 론

특정 목적을 위해 재사용이 일반적으로 수행되어 오는 동안, 더 체계적인 접근 방법을 통해 생산성과 품질의 상당한 향상이 이루어졌다. 최근 이러한 접근 방법들의 초점은 재사용 가능한 컴퍼넌트였고 이것은 CBD를 등장하게 하였다. 따라서, 컴퍼넌트 기반 개발 방법들은 재사용 가능한 멀티미디어 컴퍼넌트를 구축하고 애플리케이션을 조립하기 위하여 이들 멀티미디어 컴퍼넌트를 사용하는것에 초점을 두고 있다.

따라서, 컴퍼넌트 기반의 소프트웨어 생성을 위해 실세계의 도메인 지식을 입력으로 도메인을 분석하고 분석된 정보를 통해 hotspot을 식별하여 사용자와 개발자의 추가정보를 첨가하는 재설계 과정을 거친다. 이후 도메인에 따라 도메인 프레임워크와 애플리케이션 프레임워크를 생성한다. 이러한 컴퍼넌트 지향 방법론은 내부적으로는 컴퍼넌트/패턴 라이브러리 저장소를 통해 정보를 검색, 이해하여 추출하거나 혹은 합성으로 얻어진 정보는 각각 컴퍼넌트와 패턴에 대한 정보로 분류되고 이것은 재설계시 추가정보로 사용되어진다. 이로 인해 사용자는 멀티미디어 컴퍼넌트를 이용해 자신의 애플리케이션에 쉽게 적용가능한 개발 환경 즉, 본 논문에서는 CORBA 환경하에 컴퍼넌트/패턴 관리 시스템을 통해 멀티미디어 컴퍼넌트를 추가, 갱신, 삭제하는 기반 환경을 구축할 수 있다.

향후 연구 방향은 개발환경을 멀티미디어 컴퍼넌트에 국한하게 작업을 하였던 것을 패턴 관리로의 전환이 요구되며 이 두 가지를 통합하는 IDE (Integrated Development Environment)가 요구된다.

참 고 문 헌

[1] Backer, *CORBA Distributed Object Using Orbix*, Addison- Wesley, 1998.

[2] Moon Sel Kang, Byung-Ki Kim, "An Extends Facet Classification Scheme and Hybrid Retrival Model to Support Software Reuse", Vol. 1, No.1, KISS, pp. 23~38, 1994.

[3] Kim, Haeng-Kon, "Understanding Frameworks", Home-Coming Meeting, Sep., 1998.

[4] Component, Component Development Strategies, Vol.8, No.7, July, 1998.

[5] Philippe Kruchten, "Modeling Component Systems with the Unified Modeling Language", Available Web server from http://www.sei.cmu.edu/cbs/icse98/papers/p1.html.

[6] Dsouza D.F, and Wills A.C., *Objects, Components, and Components with UML*, Addison-Wesley, 1998.

[7] Kozaczynski Wojtek and Booch G., "Component-Based Software Engineering", IEEE Software, pp. 34~36, Sept./Oct., 1998.

[8] Sterling, "Component Based Software Development" Available by web server from http://www.sterling.com.

[9] Philippe Kruchtn, "Modeling Component System with the Unified Modeling Language", Available by web server from http://www.sei.cmu.e여/cbs/icse98/papers/, 1998 International Workshop on CBSE.

[10] Mikio Aoyama, "New Age of Software Development : How Component-Based Software Engineering Change the way of Software Development", 1998 International Workshop on Component-Based Software Engineering, 1998.

[11] Alan W, Brown and Keith Jaeger, "The Future of Enterprise Application Development with Components and Patterns", STERLING SOFTWARE, Aug., 1998.

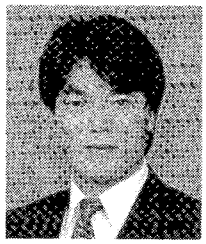
[12] 최일훈, "UML의 철저한 해부가 시작된다.", 마이크로 소프트웨어, pp. 354-362, Fab., 1999.

[13] 김행곤외 2인, "확장된 UML기반의 E-Mailing System 개발 프로세스" 정보처리학회 논문지, 제 2권 5호, pp.461-464, 1998.

[14] Microsoft Press, *Microsoft Visual C++ 6.0 Manual*, 1998.

[15] 박재현, *Core CORBA*, 영한출판사, 1998.

[16] Orfali Harkey, *JAVA and CORBA*, WILEY, 1998.



김 행 곤

1985년 중앙대학교 전자계산학과 졸업(이학사)
1987년 중앙대학교 대학원 전자계산학과 졸업(이학석사)
1991년 중앙대학교 대학원 전자계산학과 졸업(공학박사)
1988년~1989년 미 AT & T 연구원(미 시카고)
1987~1990년 한국전기통신공사 전임 연구원
1990~현재 대구효성가톨릭대학교 컴퓨터 공학과 부교수
관심분야 : 객체지향 시스템 설계, 사용자 인터페이스, 소프트웨어 재공학, 자동화 툴, CASE