설계패턴이 적용된 인트라넷 어플리케이션의 객체모델링

배 제 민[†] · 이 창 훈[††] · 이 경 환[††]

요 약

하이퍼미디어 기반 인터넷서비스의 확산으로 월드와이드웹(WWW)은 대중적으로 널리 알려지게 되었고 이런 하이퍼미디어 어플리케이션의 응용분야로서 인터넷 프로토콜인 TCP/IP와 HTTP를 이용하여 기업내부업무를 처리하고자 하는 인트라넷 어플리케이션도 점차 대두되고 있다. 인트라넷 어플리케이션은 단순한 정보획득뿐만 아니라, 정보의 생산, 수정 및 삭제 등을 하이퍼미디어 형식으로 수행해야 되기 때문에 정보의 획득에만 비중을 둔 기존의 하이퍼미디어 어플리케이션 개발방법론을 그대로 적용하기엔 부적당하다고 볼 수 있다. 본 논문은 객체지향 방법론을 이용하여 인트라넷 환경에 맞게 적용한 객체지향 인트라넷 어플리케이션 개발방법론(OOIDM)을 제안한다. 그리고 설계정보의 재사용을 위해 인트라넷 영역에서 사용될 수 있는 설계패턴을 제안하고, 이를 OOIDM에 적용한 사례연구도 제시하였다. 인트라넷 영역에서 설계패턴의 적용은 많은 잇점을 가져다 주었다. 설계패턴은 성공적인 설계정보와 그 구조의 재사용을 용이하게 해주었고, 이에 따른 설계결정(design decision)을 감소시켜 주었다.

Object Modeling of Intranet Application applying Design Patterns

Je Min Bae[†] · Chang Hoon Lee[††] · Kyung Whan Lee[††]

ABSTRACT

WWW has accepted widely to the user who wants the hypermedia-based internet services. And WWW introduces intranet environment which consists of the networks supporting TCP/IP and HTTP protocol for processing the task of company inside that. Intranet application should support not only acquiring the informations, but also producing, modifying and deleting the ones. But since previous hypermedia development methods lack modeling behavior of system and reuse, we need a new method for intranet application. In this thesis, we have proposed the OOIDM(Object Oriented Intranet application Development Method) supporting modeling behavior of system and reuse. And we have proposed the design patterns available for the intranet domain in order to reuse the design information. And we introduces a case study about OOIDM applying design patterns. Adaptation of design patterns to intranet domain gives us much benefits. Design patterns make it easier to reuse the successful designs, architecture and reducing the design decisions.

※본 논문은 정보통신연구관리단 산학연과제의 연구비지원에 의해 연구됨.

† 종신회원:중앙대학교 컴퓨터공학과
†† 정 회 원:중앙대학교 컴퓨터공학과

논문접수:1997년 2월 26일, 심사완료:1997년 6월 4일

1. 서 론

하이퍼미디어 시스템이란 멀티미디어 도큐먼트를 하이퍼텍스트형식으로 연결해주는 시스템을 의미한다[2]. 이는 쉬운 사용자 인터페이스와 복잡한 정보들을 쉽게 획득해 주는 특성때문에 주로 소프트웨어의 도움말기능이나 메뉴얼에 많이 사용되어 왔다[5]. 또한 하이퍼미디어 시스템은 정보의 생산자(author)가 의도하는 대로 정보의 소비자(reader)가 정보를 얻을 수 있는, 즉 하이퍼미디어형식의 정보획득으로 인간의 정보인지능력을 고려할 때, 인간에게 아주 친숙한 구조를 취하고 있다[4]. 이런 하이퍼미디어 시스템의 장점이 분명하게 드러나고 대중화를 이룬 것은 WWW(이하 웹)의 확산때문이다.

웹의 성공으로 인터넷 프로토콜인 TCP/IP와 HTTP를 이용하여 기업이나 단체의 업무를 처리하고자 하는 네트웍 컴퓨팅환경에 많은 사람들이 관심을 갖게 되었는데, 그것이 인트라넷(intranet)이다. 인트라넷 어플리케이션들은 웹에서 적용되는 표준적인 기술들을 그대로 사용할 수 있기 때문에 값이 저렴하며, 개방적이면서 변화하는 환경에 능동적으로 대처할 수 있게 된다.

그런데 인트라넷 어플리케이션은 하이퍼미디어 형식을 통해 기존의 어플리케이션들을 통합하여 나타나기 때문에 단순한 정보획득에 목적을 두는 것이 아니라 정보의 전달, 수정, 삭제 등의 복잡한 양상을 띄게 된다. 그러므로 정보들 사이의 하이퍼미디어 연결관계를 이용한 정보획득이란 측면뿐만 아니라 시스템의 행위적인 측면도 중시될 수 밖에 없다. 이런 인트라넷 어플리케이션이 규모가 복잡하고 커지면 복잡한 시스템의 행위로 말미암아, 비체계적인 방법으로 개발할 때 개발생산성은 물론이거니와 유지보수도 어려워질 것이다.

이런 종류의 어플리케이션을 개발하기엔 객체지향 개발방법론이 적합하다고 볼 수 있는데, 그 이유는 복잡한 실세계의 시스템을 조직화하여 모델링, 구현하기에 적합하기 때문이다[10]. 그리고 객체지향 개발방법론을 이용하여 인트라넷 어플리케이션을 모델링했을 때, 객체의 속성뿐만 아니라 서비스까지 추출하기 때문에 기존의 E-R 다이어그램을 위주로 하는 하이퍼미디어 개발방법론들에서 부족했던 인트라넷 어플리케이션이 요구하는 시스템의 행위를 모델링할 수 있는 기반을 제공한다. 객체지향 개발방법론을 통해서 얻을 수 있는 이점들은 많지만 코딩 단계에서 생성된 부품들을 재사용할 때는 그것을 자신의 개발환경에 맞게 수정을 해서 사용해야 된다. 이는 설계단계에서의 변경이 발생하면 코드자체에 많은 수정을 해야하며, 추가비용을 부담하게 하기 때문에 재사용 효율을 떨어 뜨릴 수가 있다. 그래서, 코딩 전 단계인 설계 단계의 정보를 재사용하는 개념이 설계패턴(Design Pattern)이다. 설계패턴은 특정한 문맥안에서 자주 발생하는 설계 문제들을 해결하기 위한 객체와 클래스들간의 통신 및 관계를 묘사한 것이다[11].

본 논문은 복잡한 실세계를 구현, 유지보수하는 데 효율적인 방법론인 객체지향 방법론을 이용하여 인트라넷 환경에 맞게 적용한 객체지향 인트라넷 어플리케이션 개발 방법(OOIDM, O-O Intranet application Development Method)을 제안한다. OOIDM은 웹에서 사용되는 기술요소 및 데이터들을 모델링할 수 있고 기존의 설계패턴의 개념을 인트라넷 어플리케이션 영역에 맞게 적용하여 인트라넷 어플리케이션에서 자주 발생하는 설계 정보들을 재사용하기 위한 설계패턴의 대상 및 분류, 재사용방법을 제공한다.

2. 관련 연구

2.1. 기존의 하이퍼미디어 어플리케이션 개발 방법론

하이퍼미디어 어플리케이션은 서로 다른 기술들을 가진 사람들이 협력하여 하나의 완성된 시스템을 만들고, 어플리케이션의 복잡한 정보들을 구조적으로 연결시켜서 사용자에게 분명한 정보를 제공해야 하며, 하이퍼미디어의 멀티미디어 특성을 특별히 고려해야 한다는 점에서 기존 어플리케이션과 다른 특성을 가지고 있다[6]. 기존의 하이퍼미디어 어플리케이션 방법론들은 크게 구조적분석방법론과 객체지향 개발방법론으로 나누어 진다. 구조적분석방법론은 ERD를 사용하고, 객체지향 개발방법론은 서비스를 수행하는 객체에 근거하여 모델링하는 방법이다.

(1) RMM(Relationship Management Methodology)

RMM[6]은 ERD를 기초로 한 방법론이다. ERD는

많은 시스템 분석가들에게 익숙하기 때문에 이를 근거로한 분석은 쉽게 적용될 수 있다는 장점이 있다. 그렇지만 시스템의 행위를 결정하는 단계가 개발단계 후반부에 독립적으로 존재하여 개발초기에 영역을 분석할 때 시스템행위를 모델링하기가 어렵다는 단점이 있다. 핵심적인 단계는 총 7개이며, 각각의 단계에는 지침(guideline)들이 존재한다. RMM은 주로 평범한 내부구조와 정보를 자주 갱신되어야 되는 하이퍼미디어 어플리케이션에 특히 적합하고 그렇지 않은 경우에는 별로 유용하지 않다.

(2) EORM(Enhanced O-R Model)

객체지향 방법을 하이퍼미디어 어플리케이션 영역에 적용한 연구의 결과로서 EORM[7]이란 방법론이 있다. EORM은 클래스 프레임워크(class framework), 결합 프레임워크(composition framework)와 GUI 프레임워크의 3개 프레임워크로 구성되어 있다. 클래스 프레임워크는 문제 영역에서 의미있는 클래스들을 추출하고, 각각의 클래스의 속성, 매쏘드, 일반화/상세화 관계 등을 정의한다. 그리고 결합 프레임워크는 클래스들의 링크정보들로 이루어져 있다. 링크정보들을 추출하고 그 정보들을 갱신한다. 또한 GUI 프레임워크는 클래스와 결합관계를 사용자에게 보여주기 위한 프리젠테이션(presentation)과 윈도우의 스타일을 결정한다.

(3) OOHDM(Object Oriented Hypermedia Design Model)

OOHDM[8]은 하이퍼미디어 어플리케이션 개발을 위한 객체지향 개발방법론이다. 이것은 개념적 설계(conceptual design), 네비게이션 설계(navigation design), 추상적 인터페이스 설계(abstract interface design), 구현의 단계로 이루어져 있다. 개발단계는 반복적이며 프로토타입 방식으로 수행된다. 첫 번째 단계는 개념적 설계 단계로 클래스, 클래스들 사이의 관계, 서브시스템 등을 추출한다. 추출된 클래스들을 개념 클래스(concept class)라고 지칭하고 이런 클래스들의 관계를 일반화/상세화, 집합, 일반 관계 등으로 표시한다. 두 번째 단계는 네비게이션 설계로 하이퍼미디어 어플리케이션의 특성인 네비게이션 정보를 추출하는 단계이다. 노드와 링크, 접근구조(access structure)와 같은 정보를 갖는 네비게이션 클래스들을 추출하여 똑같은 문제영역에서 사용자와 그들이 수행하는 행위에 따라 다양한 네비게이션 모델을 생성시킨다. 세 번째 단계는 추상적 인터페이스 설계로 사용자가 하이퍼미디어 어플리케이션을 통하여 직접 접하게 될 인터페이스를 정의하여 추상적 인터페이스 모델을 생성하는 단계이다. 추상적 인터페이스 모델은 하나의 네비게이션 모델로부터 여러개 이상 나올 수 있다. 사용자로부터 이벤트를 받아들일 수 있는 인터페이스 클래스를 통하여 사용자는 네비게이션을 할 수 있다. 마지막으로 네 번째 단계는 구현단계로 네비게이션, 추상적 인터페이스 모델을 구현환경에 맞춰서 구현하는 단계이다.

2.2 설계패턴(Design Pattern)

프로그래머는 일반적으로 다른 사람들이 작성해 놓은 프로그램들의 일부분을 그대로 이용하거나 변형하는 방식으로 자신의 프로그램을 작성하는 경향이 있다. 그것은 다른 사람이 이미 완성하고, 검증된 프로그램을 자신이 이용하면, 그 효과가 그대로 나타날 것이라는 기대때문이다. 이렇게 다른 전문가들이 작성한 코드나 설계의 패턴들을 모방하는 것을 추상화한 개념이 설계패턴이란 것이다. 설계패턴은 넓게 해석하면, 소프트웨어 개발영역에서, 특정한 임무를 해결하기 위해 자주 사용되는 규칙들의 집합이라고 볼 수 있다[9]. 설계패턴들은 재사용 가능한 객체지향 설계들을 생성하기에 유용한 일반적인 설계 구조를 추상화함으로써 설계 단계의 재사용방법을 도입할 수 있다. 현재까지 설계패턴이란 단어의 표준화된 의미는 존재하지 않는다. 컴퓨터뿐만 아니라 건축학 등의 다른 학문에서 오래전부터 사용되어왔고 저자에 의해서 그 의미에는 차이가 있다.

여러 설계패턴 접근 방법중에서 특히 Gamma[3] 등이 제안한 설계패턴 카탈로그의 개념이 객체지향 분야에 많은 영향을 주었다. 이 논문에서의 설계패턴이란 용어는 설계패턴 카탈로그로 간주한다. 설계패턴의 정의를 간단히 내려보면 "특정한 문맥안에서 자주 발생하는 설계 문제들을 해결하기 위한 객체와 클래스들간의 통신 및 관계를 묘사한 것"이라 할 수 있다.

또한 설계패턴은 다음과 같은 성질을 띄게 된다. 우선 소프트웨어시스템의 구조적, 기능적 원리를 구

현하기 위한 사전정의된 마이크로아키텍처를 제공한다. 그리고 현재 사용중이며, 효과가 입증된 설계 경험을 반영하며 클래스와 객체의 상위수준의 추상화에 대한 이름, 특성을 명시한다. 또한 설계 원리를 이해할 수 있는 단어역할을 하고, 소프트웨어 개발시 재사용가능한 컴포넌트의 역할을 하며, 개발영역에 의존적일 수도 있고 특정한 영역에 종속적일 수도 있다. 마지막으로 소프트웨어 설계의 기능적, 비기능적인 면을 다룰 수 있다.

설계패턴을 기술할 때 일반적으로 다음과 같은 4가지 요소를 가지고 있다 [3]. 첫 번째는 이름(name)으로 이는 설계 문제와 해결법, 결과 등을 암시하는 이름을 의미하며, 두 번째로 문제점(problem)은 문제영역을 설명하고 패턴이 적용될 시점을 기술한다. 세 번째로 해법(solution)은 설계 문제를 해결하기 위한 객체와 클래스들간의 관계(relationship), 책임(responsibility), 협동(collaboration)을 기술하고 예제코드를 명시한다. 마지막으로 결과(consequence)는 설계패턴을 적용했을 때의 장단점을 기술한다.

그렇지만 개인의 필요에 맞게 확장을 할 수도 있는데[1], 이름의 경우 패턴의 핵심을 나타낼 수 있는 이름을, 이유는 패턴을 적용하기 위한 동기를, 적용시점은 적절하게 패턴을 사용하는 시점을 나타내고, 분류는 패턴의 종류를, 설명은 패턴안에 들어 있는 클래스와 객체들간의 책임 및 관계를 설명하며 다이어그램은 패턴구조의 그래픽 표기를 나타내고 동적 행위는 패턴의 동적행위를 묘사하기 위한 텍스트 및 다이어그램을 표현한다. 또한 구축방법은 패턴을 실제적으로 구축하기 위한 단계를 기술하고 구현은 패턴을 구현하기 위한 지침을, 변종 패턴은 본 패턴의 변화된 패턴을 나타내고 예제는 패턴을 적용한 간단한 예제를 보여주며 결과는 패턴 적용시의 장단점을 설명한다. 마지막으로 이와 관련된 다른 패턴들을 알려준다.

설계패턴들을 실제로 활용하기 위해서는 패턴들이 패턴저장소(repository)에 체계적인 분류가 되어, 개발자들이 쉽게 원하는 패턴을 찾는 과정이 필요하다. 그리고 발견한 패턴을 자신의 구현환경에 맞춰서 패턴안에 있는 클래스들을 구현환경에 대응되는 클래스들로 변환시키는 과정이 필요하다. 패턴들을 분류할 때, 그 기준은 추상화의 수준(granularity), 기능성(functionality), 구조 원칙(structural principle) 등이 될 수 있다[1].

그 중 추상화의 수준에 따라 분류한 내용을 살펴보면, 아키텍처 프레임워크는 소프트웨어 시스템을 구성할 때, 가장 기본적인 골격을 표현하는 패턴(예: MVC 프레임워크)을 나타내고, 설계패턴 카탈로그는 소프트웨어 아키텍처의 컴포넌트나 서브시스템을 구성하기 위한 마이크로 아키텍처를 의미하며, 이디엄(idiom)의 경우에는 패턴의 특정한 컴포넌트를 구현하는 방법, 기능, 관계 등을 묘사한다. 이는 특정 프로그래밍 언어에 종속될 수 있다.

위의 아키텍처 프레임워크, 설계패턴 카탈로그에 속한 패턴들은 다시 2개의 타입으로 분류할 수 있는데, 첫 번째는 일반적 패턴(general pattern)으로 어떤 영역에도 적용할 수 있는 패턴을 의미하고 영역종속적 패턴(domain-specific pattern)은 특정 영역(어플리케이션의 영역, 프로그래밍언어 등)에서 적용될 수 있는 패턴을 나타낸다.

또한 설계패턴들을 개발시 적용하여 얻을 수 있는 효과는 다음과 같다. 우선 소프트웨어 아키텍처의 폭넓은 재사용을 지원하고, 소프트웨어 개발팀 내부나 다른 팀사이와의 분명한 의사전달을 가능하게 해주며, 새로운 개발자들에게 교육의 역할을 한다. 그리고 현존하는 개발방법론의 설계 경험을 반영할 수 없는 약점을 보완해주고, 결론적으로 소프트웨어 설계의 만국어 역할을 한다. 이런 장점에도 불구하고, 설계패턴은 상당히 경험적, 주관적이고, 기존 개발방법론에 부드럽게 적용되어 통합되지 못한다는 점과 지원 케이스툴의 부족 등의 문제점들이 있다. 그렇지만, 모든 분야, 상황에 설계패턴을 강제적으로 적용하는 것이 아니라, 검증된 설계패턴이 효과를 낼 수 있는 상황에 적용했을 때 좋은 효과를 볼 수 있는 것은 이미 인정된 사실이다.

본 논문에서는 기존의 객체지향 개발방법론에서 부족했던, 다른 전문가들의 개발지식을 활용할 수 있는 역할로서의 설계패턴을 활용한다. 인트라넷 어플리케이션은 여러 분야(내용물, 프로그래밍, 인터페이스 등)의 기술이 사용되고, 사용자의 인지적 측면과 개발자의 경험적인 면을 적극적으로 반영해야 하기에, 다른 전문가들의 이미 검증된 설계패턴의 활용이 더욱 중요하다.

3. OOIDM(O-O Intranet Application Development Methodology)

OOIDM은 기본적으로 OOHDM(Object Oriented Hypermedia Design Model)[8]의 라이프사이클을 인용한다. 그렇지만, 인트라넷 어플리케이션을 모델링하기 위해 필요한 시스템의 행위를 추출하고 반영하기 위해, 각 개발단계에서 추출되는 클래스들의 의미가 다르다. 또한 시스템 행위를 구체적으로 설계하기 위한 "실행가능 설계"라는 개발단계가 추가되었다. OOIDM은 (그림 1)과 같은 개발단계를 갖는다.

각 개발단계를 통해 클래스들이 추출되며 최종 구현단계에서는 인터페이스 클래스와 실행가능 클래스들이 구현된다. OOIDM에서는 인트라넷 어플리케이션이 제공하는 모든 서비스들은 특정 서비스를 수행하는 인트라넷 객체들과 협력하여 서비스를 제공하게 되고 그런 객체들은 하이퍼미디어나 프로세스 형태로 사용자들에게 나타나게 된다.

3.1 영역분석(Domain Analysis) 단계

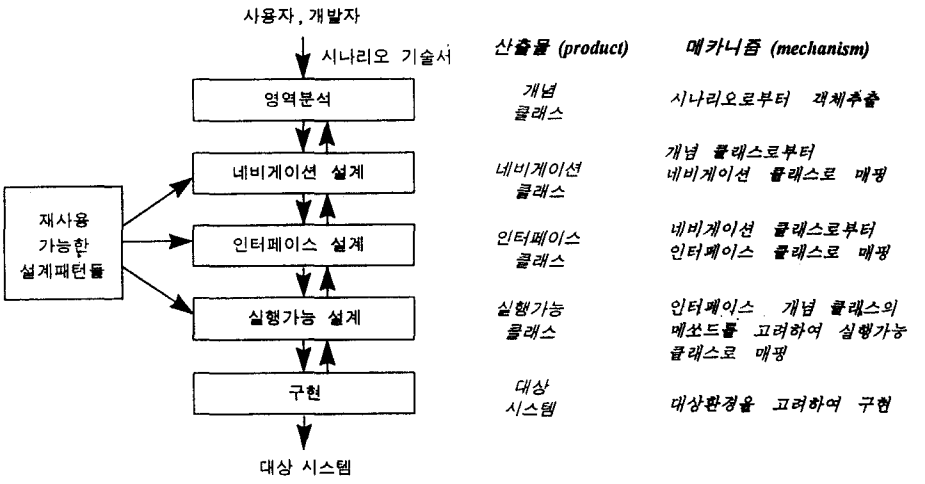
본 단계는 사용자로부터 요구조건문과 시나리오 기술서를 받아서 실세계에 존재하는 객체를 추출하는 과정이다. 시나리오(scenario)란 사용자가 시스템을 사용하게 될 흐름대로 사용요청을 기술한 것이다. 시나리오를 통해 사용자의 관점에서 시스템의 행위를 표현할 수 있고, 개발의 초기단계에서부터 사용자가 참여할 수 있으며, 요구명세의 타당성을 검증하는 기반으로 사용할 수 있다.

시나리오 작성을 위해서는 먼저 시스템의 목적과 목표를 분명히 파악하고 이에 부응하는 사용자 그룹별 시나리오를 기술한다. 이런 사용자 그룹별 시나리오를 통해서 시스템의 서비스를 추출할 수 있다. 하이퍼미디어 시스템은 사용자의 시나리오를 통해서 하이퍼링크의 네비게이션 정보와 사용자 인터페이스를 결정할 수 있기 때문에 다른 시스템보다 시나리오를 통한 요구분석이 더욱 중요하다.

시나리오 기술서에는 문제영역의 클래스와 속성, 메쏘드(method), 클래스들 사이의 결합관계(association)와 집합관계(aggregation), 일반화/상세화 관계 등이 존재한다. 먼저 시스템을 사용할 사용자들을 분류한 후에 각 사용자들이 시스템을 사용하는 시나리오를 기본과 확장 시나리오 기술서를 통해 기술한다.

기본 시나리오 기술서에는 시스템 사용자 그룹, 사용자 역할, 사용목적, 사용자 정보, 시스템 반응, 시스템 서비스, 시스템 처리 등이 기술된다. 시스템 처리는 시스템이 서비스를 수행하기 위한 내부행위를 의미한다.

만약 기본 시나리오 기술서의 한 사용목적을 이루



(그림 1) OOIDM의 라이프사이클
(Fig. 1) Lifecycle of OOIDM

기 위한 행위가 많을 때 그것을 하나의 기술서로 독립해서 기록할 수가 있는데, 그것을 확장 시나리오 기술서라 한다. 확장시나리오 기술서의 구성요소는 기본 시나리오 기술서와 같다. 사용자들이 서브시스템들을 독립적으로 사용할 경우에는 각 서브시스템에 대해 시나리오 기술서를 작성한다. 시나리오 기술서는 시스템에 대한 요청과 응답을 중심으로 기술한다. 시나리오 기술서로부터 추출해야 하는 클래스와 그들간의 관계를 결정하는 단계는 다음과 같은 순서로 한다.

①시나리오 기술서에서 주어, 목적어 등의 명사들은 후보 객체 또는 객체의 후보 속성으로 취급한다. 또한 서술어 등의 동사들은 해당 객체의 후보 메쏘드로 취급한다.

②후보 클래스, 속성, 메쏘드들 중 어플리케이션내에서 의미적으로 부적절한 것들은 버리고, 확실한 클래스를 결정한다. OMT[10]의 객체모델에서 부적절한 클래스, 속성, 메쏘드를 파악하는 기준을 그대로 적용한다.

③클래스들사이의 일반화/상세화, 집합관계, 결합관계 등을 추출한다.

④위의 3과정을 반복하여, 개선해 나간다.

⑤OMT의 객체모델에 사용되는 표기법으로 객체모델을 작성한다. OMT의 객체모델을 선택한 이유는 OMT방법론이 학계 및 산업계에서 널리 사용되어 개발자들에게 친숙하기 때문이다.

⑥시스템 문맥 다이어그램(context diagram)을 작성한다. 시스템 문맥 다이어그램은 선택가능하며 사용자 종류별로 시스템에 대한 요청과 응답을 다이어그램으로 표시하는 것이다.

이 단계를 거쳐서 추출되는 클래스들은 개념 클래스와 실행가능 클래스로 나누어진다. 개념 클래스는 구현되어질 때 사용자에게 직접적으로 나타나는 클래스로서, HTML형태로 제공하는 서비스의 기간이 되는 클래스이다. 즉 하이퍼미디어 어플리케이션 영역에서 의미있는, 하이퍼미디어형태의 정보를 보여주고 서비스할 수 있는 개념, 사물, 장소를 의미한다. 이 클래스의 속성은 사용자에게 보여지는 정보를 제공할 소스데이타이다. 메쏘드는 해당 객체의 서비스를 의미하며, 사용자로부터 정보를 획득, 전송, 수정, 삭제 등의 역할을 하게 된다. 이런 메쏘드 중 일부는 나중 단계에서 "실행가능 클래스"로 대응될 수도 있다.

실행가능 클래스는 사용자에게 직접적으로 나타나지 않는, 클라이언트나 서버의 프로세서를 점유해서 서비스를 제공하는 클래스로서, 나중에 "실행가능 설계" 단계에서 구체적인 내부가 결정이 된다. 예를 들면, 웹을 통해 책을 주문하는 시스템에서 주문자는 자신의 정보(이름, 신용카드번호 등)와 주문한 책의 이름 등을 서버에게 보내게 된다. 서버는 전달받은 정보를 CGI로 처리하여 책주문을 받는다. 이때의 CGI 프로그램이 객체지향 언어로 작성된 경우 구성된 클래스를 실행가능 클래스라 한다. 그러므로 이 단계에서는 개념 클래스가 추후에 확인되는 실행가능 클래스보다 더욱 중요하고 추후 개발단계의 기간이 되는 클래스로서의 역할을 한다.

3.2 네비게이션 설계(Navigation Design)단계

하이퍼미디어 어플리케이션에만 있는 설계 단계로서 사용자가 하이퍼미디어 방식으로 원하는 정보를 접근, 서비스를 수행하기 위한 네비게이션 정보(노드, 링크, 앵커, 접근구조 등)를 추출한다. 네비게이션 정보를 추출하기 위해, 시나리오 기술서에 나타나 있는 시스템에 대한 요청과 획득에 대한 정보인 사용자 정보, 시스템 반응, 시스템 서비스를 참고한다.

네비게이션 정보를 표현하기 위해 사용되는 용어[4] 중 노드는 사용자에게 보여주는 정보의 단위로서 인트라넷 어플리케이션에서는 HTML이나 멀티미디어 파일로서 존재한다. 링크는 연관된 다른 노드들을 연결시켜주는 노드안에 있는 연결선이고 앵커는 링크의 출발 위치로서 노드내에 단어나 버튼으로 나타난다. 사용자가 앵커를 선택했을 때 네비게이션이 시작된다.

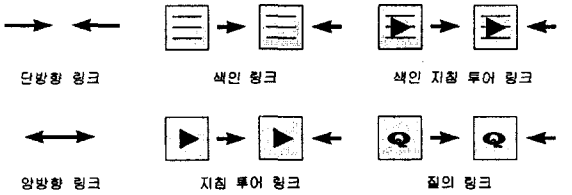
추출된 개념 클래스들로부터, 그들끼리의 서비스를 수행하기 위한 네비게이션 정보, 즉 개념 클래스들 사이에 링크와 접근구조 정보를 추상화한 네비게이션 클래스를 추출한다. 예를 들어, 수강시스템에서 교수와 학생이라는 개념 클래스가 추출되었고 사용자가 교수의 정보를 보고 있을 때, 그가 가르치는 학생의 정보를 보고자 할 경우 네비게이션을 해야 한다. 네비게이션을 하기위해서는 둘 사이에 네비게이션 클래스 안에 네비게이션 정보를 정의해야 한다.

네비게이션 클래스의 속성은 개념 클래스들 사이의 네비게이션 정보이며 개념 클래스들 사이의 하이퍼미디어 접근방식을 메쏘드로 표현한다. 개념 클래스를 하나의 추상화된 노드라고 가정하고 앵커(anchor)와 대상(target)과 접근 방식(access primitive) 등을 네비게이션 클래스안의 메쏘드로 표현하게 된다.

앵커는 한 개념 클래스에서 다른 개념 클래스로 연결되는 출발위치이고, 대상은 앵커를 사용자가 선택했을 때 획득하게 되는 정보를 의미한다. 접근 방식은 앵커를 통해 대상 정보를 얻는 방식을 의미한다. 네비게이션 클래스안의 메쏘드의 형식은 [method_id] *anchor_name method_name access_primitive* target _name과 같다.

method_id는 메쏘드의 순서로서 인터페이스 클래스에서 사용된다. anchor_name은 HC.앵커이름 또는 HC.앵커이름1, HC.앵커이름2,... 으로 표기되고, HC란 앵커나 대상이 속할 개념 클래스 이름을 의미한다. method_name은 개념 클래스사이의 관계이름을 이용하여 본 접근을 통하여 얻을 수 있는 정보를 상징하는 이름을 선택하면 된다. target_name은 HC 또는 HC.대상이름 또는 HC.대상이름1, HC.대상이름2,... 으로 표기된다. access_primitive는 다음과 같이 7가지[6, 13]가 있으며 자신의 개발환경에 따라 추가할 수 있다.

U는 단방향 링크(Unidirectional link)로 다른 노드로 가는 앵커가 한 쪽에만 있는 경우이고, B는 양방향 링크(Bidirectional link)를 의미하며 두 노드에 양쪽과 연결된 앵커가 있는 경우이다. I는 색인 링크(Index link)로 다른 노드들의 색인이 링크로 연결되어 있음을 나타내고, T는 지침 투어 링크(guided Tour link)로 다른 노드들을 순차적으로 접근할 수 있는 링크를 나타낸다. IT는 색인 지침 투어 링크(Indexed guided Tour link)로 인덱스 링크와 지침 튜어링크의 혼합이고, Q는 질의(query) 링크로 사용자가 입력한 데이타를 바탕으로 정보를 획득하는 링크이며, G는 그룹(group) 링크로 여러 개념 클래스가 대상대상이 될 때 사용되며 각각을 앵커로 연결한다. 각각의 접근방식에는 (그림 2)와 같은 기호를 사용할 수 있다.



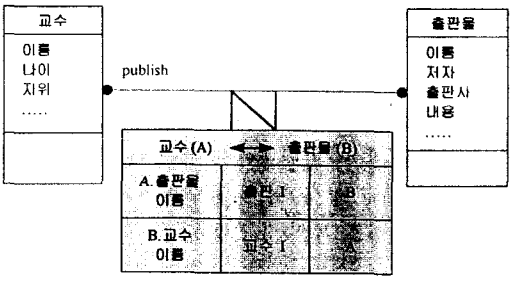
(그림 2) 기본접근방식에 대한 기호들
(Fig. 2) Symbols about access primitives

이런 접근방식을 선택할 때는, 사용자의 인지적 측면과 개념 클래스들사이의 일반화/상세화, 집합 관계 등을 고려해야 한다. 이에 대한 지침은 〈표 1〉과 같다.

〈표 1〉 개념 클래스 관점에 따른 연결관계
〈Table 1〉 Link relationships in terms of concept class

개념 클래스사이의 관계	접근 방식
1:N관계	단방향 인덱스 링크
M:N관계	양방향 인덱스 링크
집합관계	그룹링크
일반화/상세화 관계	그룹링크

예를 들어 대학정보 인트라넷 어플리케이션에서 사용자가 특정교수가 저자인 출판물에 대한 정보를 얻고자 한다. 그리고 한 출판물은 여러 교수가 저작할 수 있고 한 교수는 여러 출판물을 저작할 수 있다고 가정한다. 이때 교수와 출판물이라는 두 개의 개념 클래스가 추출이 되고, 그 둘사이에는 다대다(M:N)관계가 성립한다. 만약 사용자가 특정교수의 정보를 HTML형태로 보고 있을때, 해당교수가 저자인 출판물 정보를 획득하기 위한 접근방식으로 교수가 지은 여러 출판물을 가리키는 앵커를 통한 색인링크방

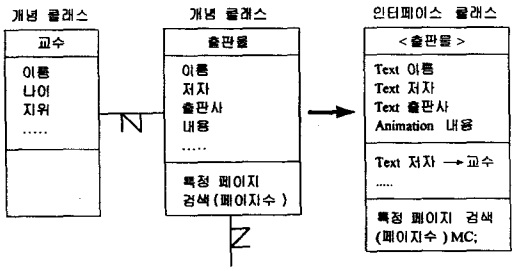


(그림 3) 교수, 출판물에 대한 네비게이션 클래스
(Fig. 3) Navigation class about professor and publisher

식을 선택한다. 또한 출판물 정보를 보고 있을 경우, 그것의 저자에 대한 정보를 얻기를 원할 경우에도, 여러 교수들을 가리키는 앵커를 설정하기 위해, 색인 링크 방식을 선택한다. 이에 대한 하이퍼미디어 클래스와 네비게이션 클래스는 (그림 3)과 같다. 기호 N은 OMT에 추가된 표기법으로 두 개의 개념 클래스간의 네비게이션 정보를 갖고 있는 클래스를 나타낸다.

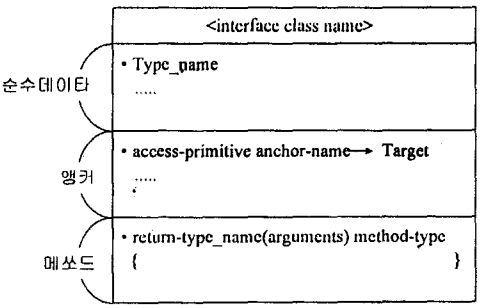
3.3 인터페이스 설계(Interface Design) 단계

인터페이스 설계 단계는 추출된 개념 클래스와 네비게이션 클래스를 바탕으로, 사용자가 인트라넷 어플리케이션을 직접 사용할 때 접하는 인터페이스 역할을 하는 인터페이스 클래스를 추출하는 단계이다. 즉 인트라넷 어플리케이션으로 최종적으로 구현되었을 때의 노드들로 사용자에게 나타나게 되는데 (그림 6)과 같이 개념 클래스와 네비게이션 클래스, 실행가능 클래스를 이어주는 역할을 하게 된다.



(그림 4) 개념 클래스가 인터페이스 클래스로 대응되는 과정
(Fig. 4) Process of mapping concept class to interface class

(그림 4)는 개념 클래스가 네비게이션 클래스의 하이퍼링크 정보를 바탕으로 인터페이스 클래스로 바뀌는 과정을 보여주고 있다. 이 과정에서는 개념 클래스의 속성들에 대한 특정 타입이 결정되고 네비게이션 클래스의 링크 정보가 인터페이스 클래스의 앵커속성에 추가된다. 인터페이스 클래스의 표기법은 (그림 5)에 나타나 있다. 즉, 인터페이스 클래스는 사용자가 브라우저를 통하여 접하게 될 하이퍼미디어 정보를 나타내는 클래스로서, 여러 HTML파일과 멀티미디어파일, 즉 여러 노드로 최종적으로 구현된다. 한 개념 클래스는 최소한 1개의 인터페이스 클래스가 존재하게 된다. 만약 한 개념 클래스의 정보를 여러 사용자의 관점에 따라 다른 인터페이스가 필요할 땐, 사용자의 종류에 따라 여러 인터페이스 클래스가 존재할 수 있다.



(그림 5) 인터페이스 클래스의 표기법
(Fig. 5) Notation of interface class

인터페이스 클래스의 속성은 개념 클래스의 속성을 대응시킨 것으로서 크게 2가지 종류로 구성된다. 순수데이터의 경우 사용자에게 직접적으로 보여질 특정미디어나 동적인 효과를 나타내며, Type_name은 속성 이름을 나타내고 그 타입은 Text, Image, Motion, Sound 등과 같다. 앵커는 네비게이션 클래스의 앵커정보를 접근 방식 기호로 표시하며 그 형식은 Anchor_num Anchor_type Anchor_name Acess_primitive Target_name와 같으며 Anchor_num은 앵커의 id를 나타내는 숫자이고, Anchor_type은 Text, Image, Motion, Sound 등을 의미하며, Acess_primitive는 U, B, I, T, IT, Q, G 또는 해당기호를 나타낸다. 앵커이름과 대상이름은 네비게이션 클래스에서 사용되는 것과 같은 용례로 사용하면 된다.

인터페이스 클래스의 메쏘드 또한 2가지 종류가 있다. 독립적 서비스는 하이퍼미디어 클래스의 메쏘드로서 독립적인 서비스를 제공하고 CGI, MC(Java Applet, ActiveX control, 각종 스크립트 언어 등)로 구현된다. 또한 나중에 "실행가능 설계"단계에서 실행가능 클래스로 대응된다. 형식은 Return_type Method_name(argument list,..) Method_type와 같다.

Method_type은 FORM-CGI, FORMLESS-CGI, MC, MC/APP 등이며, FORM-CGI는 HTML의 형식(form)을 이용해서 서버쪽에서 실행되는 프로그램이고, FORMLESS-CGI는 형식을 이용하지 않고 서

버쪽에서 실행되는 프로그램을 의미한다. MC(Mobile Code)는 네트워크를 통해 서버쪽에서 클라이언트쪽으로 이동해서 실행되는 프로그램으로서 독립적으로 실행된다. MC/APP는 다른 서버쪽에 있는 어플리케이션과 연결되어 동작하는 MC를 의미한다.

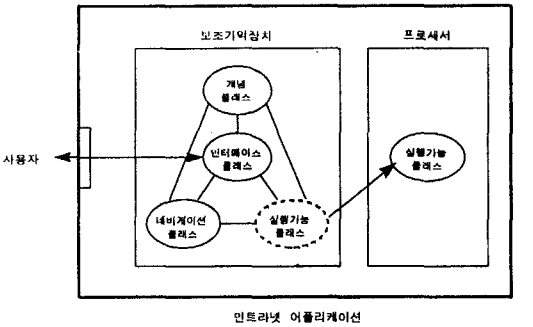
비독립적 서비스는 독자적으로 서비스하지 못하고, 하이퍼링크로 연결된 다른 하이퍼미디어 클래스의 메쏘드를 실행하도록 연결해주는 메쏘드로 앵커로 구현되며 형식과 표기법은 다음과 같다.

```
void method_name
{anchor_num;//앵커의 번호
concept_class_name.method_name();//다른 개념 클래스의 메쏘드 이름
....
}
```

네비게이션 설계단계가 두 개념 클래스들 사이의 네비게이션 정보를 추상화함으로써 하이퍼미디어 시스템의 특성인 하이퍼정보를 추출하는 단계인 반면, 인터페이스 설계단계는 실제로 사용자가 컴퓨터 모니터상에서 접하게 되는 미디어와 서비스의 구체적 표현을 설계하는 단계란 점에서 차이가 있다. 따라서 두 설계 단계는 상당히 독립적인 단계이다. 이 두 단계를 나눔으로써 하이퍼미디어 어플리케이션의 독특한 특성인 네비게이션 정보를 인터페이스와 독립적으로 설계할 수가 있다. 그러므로 한 네비게이션 설계 결과물로부터 여러개의 인터페이스 설계을 할 수 있기 때문에 네비게이션 정보는 변하지 않고, 사용자 인터페이스만 달라지는 시스템의 변동사항이 있을 때 인터페이스 설계만 새롭게 작성하여 시스템을 유지보수할 수 있는 확장성을 제공하게 된다.

3.4 실행가능 클래스 설계(Executable Class Design) 단계

인트라넷 어플리케이션의 행위를 구체적으로 설계하는 단계로서 인터페이스 클래스와 네비게이션 클래스를 입력으로 받아서 인터페이스 클래스의 메쏘드와 속성중에서 프로세스가 필요한 부분을 실제 시스템의 메모리와 프로세서를 점유할 수 있는 실행가능 클래스의 메쏘드로 매핑한다. 이전 단계의 추출된 클래스와의 관계는 (그림 6)에서 보여주고 있다.



(그림 6) 실행가능 클래스와 다른 클래스들과의 관계
(Fig. 6) Relationship between executable class and other classes

즉 실행가능 클래스는 인트라넷 환경에서 CGI 프로그램이나 자바애플릿이나 스크립트 언어와 같은 이동식 코드로 구현되게 된다. CGI와 MC가 객체지향 언어로 구현되어질 때 실행가능 클래스는 쉽게 구현될 수 있으나, 객체지향 언어가 아닌 경우는 실행가능 클래스의 속성과 메쏘드를 가상적으로 구현해야 된다. 그러므로 본 방법론으로 어플리케이션을 개발할 경우 CGI나 MC는 객체지향 언어로 구현하는 것이 바람직하다.

실행가능 클래스들의 메쏘드로 대응되는 인터페이스 클래스의 요소들은 인터페이스 클래스의 메쏘드 중에서 FORM-CGI, FORMLESS-CGI, MC, MC/APP와 인터페이스 클래스의 앵커 중에서 접근 방식이 질의(query)타입이거나 개념 클래스의 특정정보를 보여주기 위해 프로세스가 가공해서 사용자에게 공급할 필요가 있는 것, 그리고 인터페이스 클래스의 속성 중에서 텍스트스크롤, 애니메이션 동과 같은 특수효과가 필요한 부분은 MC화, 그리고 "영역분석" 단계에서 추출된 클래스 중 개념 클래스가 아닌 클래스는 본 단계에서 실행가능 클래스가 될 수 있다.

실행가능 클래스의 핵심서비스는 시나리오 기술서의 시스템처리부분을 고려하여 결정한다.

만약 이런 대응되는 실행가능 클래스들이 복잡하게 연결되어 하나의 시스템을 구성할 경우, OMT의 방법론을 적용하여 다시 객체들을 추출할 수 있다. 객체지향 언어로 개발할 경우에는 후일의 재사용을

고려한 설계과정이 필요하다. 표기법은 OMT방법론의 객체모델에서 사용되는 방식을 적용한다.

3.5 구현단계

인트라넷 어플리케이션의 구현대상인 하드웨어, 소프트웨어 환경에 맞게 구현한다. 입력으로 인터페이스 클래스와 실행가능 클래스가 있는데, 인터페이스 클래스는 HTML과 각종 미디어로 구현하고, 그것들을 화면에 어떻게 표현할지 화면배치 및 링크를 결정한다. 실행가능 클래스는 구현환경에 맞는 CGI와 MC로 구현한다. 구현후, 사용자 시나리오 기술서에 바탕하여 테스트를 한다.

4. OOIDM과 설계패턴

인트라넷 어플리케이션은 서버쪽 구현과 클라이언트쪽 구현으로 나눌 수 있다. 서버쪽은 주로 CGI프로그램이나 일반프로그래밍 언어로 작성된 어플리케이션이 동작하게 되고, 클라이언트 쪽은 HTML 및 MC들, 즉 자바 애플릿(Java Applet)이나 각종 스크립트 언어가 실행되며 이런 것들이 서로 연합하여 하나의 완성된 어플리케이션을 이루게 된다. 개발자들은 이런 인트라넷 어플리케이션에 쉽게 적용될 수 있는 설계패턴을 정의, 재사용함으로써 생산성 및 유지보수성을 향상시킬 수 있다.

4.1 설계패턴의 대상

기존의 설계패턴들은 그것을 시스템에 구현할 때의 수단이 C, C++ 같은 프로그래밍 언어로서 동적인 프로세스의 형태로 구현되었다. 그렇지만 인트라넷 환경에서의 설계패턴들은 구현 수단을 프로그래밍 언어로만 국한시킬 수가 없는데, 그것은 인트라넷 어플리케이션이 정적인 정보와 동적인 프로세스가 결합된 형태로 구현되기 때문이다. 즉 정적인 정보들은 HTML과 관련 미디어 파일로 구현되고, 동적인 프로세스들은 서버측에서는 CGI 프로그램 및 일반 어플리케이션, 클라이언트 쪽은 자바 애플릿(java applet)과 같은 이동식 코드로서 구현되기 때문에, 설계패턴의 구현 대상을 재정의해야 한다. OOIDM에서 추출되는 클래스로 설계패턴을 표현할 때 다음과 같은 3부류로 대상을 나눌 수 있다.

① 하이퍼미디어 클래스와 네비게이션 클래스의 관계 및 상호동작은 클라이언트 쪽에 하이퍼링크 정보를 포함한 HTML파일과 다른 미디어 파일로 구현

② 개념 클래스, 네비게이션 클래스, 실행가능 클래스의 관계 및 상호동작은 클라이언트 쪽에 하이퍼링크 정보를 포함한 HTML파일과 다른 미디어 파일들과 서버쪽에 있는 CGI 프로그램 및 일반 어플리케이션으로 구현

③ 실행가능 클래스들의 관계 및 상호동작은 서버쪽에 있는 CGI 프로그램 및 일반 어플리케이션으로 구현

4.2 설계패턴의 분류

위의 3가지 단위로 표현되는 설계패턴들은 개발자가 자신의 개발환경에 맞는 설계패턴을 선택하여 이용하는데 도움을 주기 위하여 2가지 기준으로 분류할 수 있다.

(1) 추상화 정도(abstraction level)는 아키텍처 프레임워크(architectural framework) 패턴의 경우 가장 높은 수준의 추상화정보를 가지고 있으며 어플리케이션 전체 아키텍처를 묘사하는 패턴이다. 또한 설계패턴 카탈로그(Design Pattern Catalogs) 패턴은 중간단계의 추상화정보를 가지고 있으며, 전체 아키텍처의 부분을 묘사하는 패턴으로서, 마이크로아키텍처 형태로 구현될 수 있다. 이디엄(Idiom) 패턴은 제일 낮은 단계의 추상화정보를 가지고 있으며, 웹 서버의 내용구축을 위한 패턴들[8], 사용자 인터페이스를 결정하는 패턴들, MC언어에 종속적인 패턴들, 설계패턴을 특정 프로그래밍언어로 시스템에 어떻게 구현할지를 묘사하는 패턴들[12]로 구성된다.

아키텍처 프레임워크와 설계패턴 카탈로그는 다시 일반, 특수 패턴으로 나누어 진다. 일반 패턴은 어떤 어플리케이션에도 적용될 수 있는 폭넓은 패턴을 의미하고, 특수 패턴은 특정 어플리케이션, 즉 GUI나 DB 등에 한정되어 적용되는 패턴을 의미한다. 여기서는 아키텍처 프레임워크와 설계패턴 카탈로그식 패턴과 서버/클라이언트 패턴에 대해서 중점적으로 다룰 것이며, 설계패턴은 이름, 동기, 적용시점, 분류, 구조도, 구조 설명, 동적 행위 설명, 구현사례 순으로 기술할 것이다.

(2) 구현 타입의 경우 서버 패턴은 CGI 프로그램 및

서버쪽의 일반 어플리케이션으로 구현됨을 나타내고 클라이언트 패턴은 HTML 및 각종 미디어 파일 및 MC로 구현된다. 또한 서버/클라이언트 패턴은 서버쪽의 CGI 프로그램 및 일반어플리케이션과 결합된 HTML및 MC로 구현된다.

4.3 설계패턴의 적용

객체가 다른 객체를 포함하고 관리하는 Container 패턴은 다음과 같이 정의된다.

①이름:Container 패턴

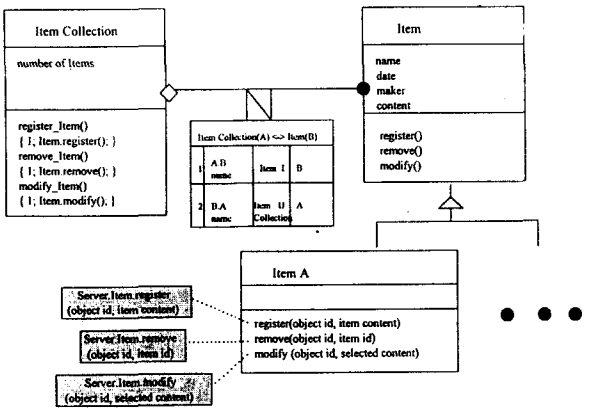
②동기:한 객체가 다른 객체들을 포함하고, 그 관리를 책임지게 함으로써 포함된 객체들에 대해서 일괄적인 관리가 가능하다. 예를 들면, "도서관리시스템"에서는 "도서장부"는 "도서"를 포함하고 그 관리를 책임지는 관계로 볼 수 있다.

③적용시점:한 객체가 다른 객체를 포함하고, 그 관리를 책임질 때

④분류:설계패턴 카탈로그 방식, 클라이언트/서버 방식

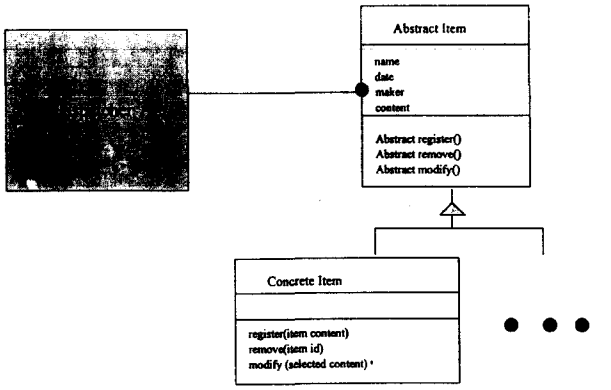
⑤구조도

*클라이언트



⑥구조 설명:클라이언트 (개념 클래스 및 네비게이션 클래스 들)의 경우 Item Collection은 Item을 포함하는 클래스로서 Item을 관리하는 인터페이스를 제공하고 Item을 관리하는 메쏘드를 갖고 있다. Abstract Item은 추상 클래스로서 실질적으로 구현될 Item에 대한 인터페이스 역할을 한다. Concrete Item

*서버

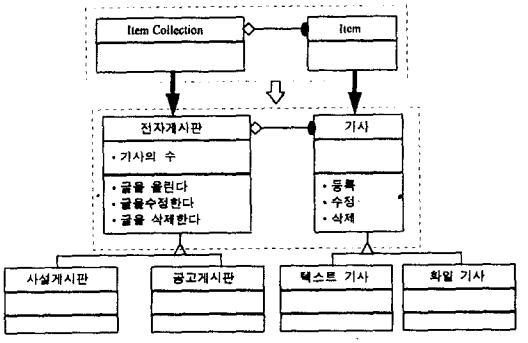


은 Abstract Item으로부터 상속받은 클래스로서 구현되는 Item이고 아이템을 등록, 삭제, 수정할 수 있는 메쏘드를 가지고 있으며, 구체적인 행위를 정의한다. Item Collection〈-〉Item은 Item Collection과 Item사이의 네비게이션 정보를 가지고 있는 네비게이션 클래스로서 인덱스 방식의 접근 방식을 취한다. 서버(실행가능 클래스들)의 경우는 Abstract Item이 등록될 Item에 대한 인터페이스를 담당하는 추상 클래스로 클라이언트의 개념 클래스인 Abstact Item과 대응되고 Concrete Item은 등록될 아이템에 대응되는 실제로 구현될 클래스로서 아이템의 등록, 수정, 삭제의 행위를 정의하고 클라이언트의 개념 클래스인 Concrete Item과 대응된다.

⑦동적 행위 설명:사용자는 Item Collection 클래스로부터 Item 클래스의 객체를 관리하게 된다.]아이템을 등록하기 위해, Item Collection의 "register_item" 메쏘드를 실행하면, 즉 Item Collection을 구현한 HTML 문서에서 "register_item"의 서비스에 대응되는 앵커를 선택하면, Concrete Item 클래스의 "register"메쏘드를 수행할 수 있게 된다. 사용자가 Concrete Item 클래스의 "register"메쏘드를 수행하면, 등록대상의 정보와 "register"메쏘드가 실행되는 Concrete Item 클래스의 정보를 서버쪽에 보낸다. 서버쪽에서는 클라이언트쪽에서 보낸 등록 정보와 메쏘드 실행 클래스 정보를 받아서, 서버쪽에 해당되는 Concrete Item 클래스의 객체를 찾아 "register"라는 메쏘드를 수행한다.

⑧구현사례:적용 어플리케이션의 경우 전자게시

판 어플리케이션에서 전자게시판과 그곳에 실리는 기사는 (그림 7)과 같이 각각 Item Collection과 Item에 대응된다. 전자게시판이 등록되는 기사들을 통합적으로 관리하기 때문이다. 또한 "Item Collection"의 "등록한다"란 메쏘드가 "보드"의 "글을 올린다"라는 메쏘드와 대응한다. 등록되는 기사들은 브라우저의 화면에 입력하는 텍스트와 클라이언트의 하드디스크에 있는 텍스트 파일 등 2부류로 나눌 수 있다. 이때, 기사는 Abstract Item과 텍스트 기사와 파일 기사는 Abstract Item을 상속받은 Concrete Item과 대응될 수 있다.



(그림 7) Container 패턴을 적용한 예
(Fig. 7) Example of applying Container pattern

Item Collection에서 "register_item"이란 메쏘드를 실행한다는 것은, 하이퍼링크로 연결된 Concrete Item의 "register"메쏘드를 실행하는 것이다. 만약 사용자가 입력하는 텍스트 글을 등록할 경우에는, Concret Item이 "텍스트 기사"라는 클래스로 대응되면서 사용자의 이벤트에 의해 "register"가 실행되면서, 서버쪽에 있는 "텍스트 기사"라는 클래스의 객체의 "register"라는 메쏘드를 실행하여, 전자게시판에 기사를 등록하게 된다. 또한 파일 기사를 등록할 경우에는 Concrete Item이 "파일 기사"가 되며, 마찬가지로 사용자의 이벤트를 받아 "register"메쏘드를 실행하여 서버쪽의 "파일 기사"라는 클래스 객체를 실행하여, 전자게시판에 기사를 등록한다.

서버의 경우는 클라이언트로부터 서버쪽에 있는 Concrete Item객체의 메쏘드를 실행하라는 요청이 들어오면, Dispatcher가 해당되는 Concrete Item객체를 생성시키고 그 객체의 메쏘드를 실행시킨다. Dispatcher는 CGI프로그램이나 일반 어플리케이션이다. 이때 Dispatcher는 Concrete Item객체가 Abstract Item객체를 상속받았기 때문에, Concrete Item객체를 Abstract Item 클래스타입으로 객체를 생성시켜서 Abstract Item객체의 인터페이스로 메쏘드를 실행한다. 이것은 폴리모피즘의 효과로서, Dispatcher가 Concrete Item의 구체적인 행위를 몰라도 해당 객체의 메쏘드를 실행시킬 수 있는 장점이 생긴다. 만약 클라이언트에서 "텍스트 기사"란 개념 클래스에서 "register" 메쏘드를 실행하면, 서버쪽에서는 "텍스트기사" 클래스의 객체를 "기사"라는 클래스 타입으로 생성시켜서, "기사"객체의 "register"를 실행한다. 물론 "기사"객체의 실질적인 타입은 "텍스트기사"이기 때문에, 텍스트기사를 등록할 수 있는 행위가 실행된다.

Dispatcher는 패스트트랙 웹서버의 서버쪽의 자바 프로그램을 실행시키는 게이트웨이 프로그램으로서, 부분적인 자바언어 소스 코드를 소개하면 다음과 같다.

```
import TextArticle;
import FileArticle;
import Article;
....
class registerArticle extends HttpApplet{
Article new_article;
public void run( ) {
    class_name = getFormField("class_name");
    new_article = new (Article)Class.forName(class_name).
    newInstance( );// TextArticle이
나 FileArticle을 생성한다.
    new_artcle.register( );// 해당 객체의 register실행
}
```

Abstract Item 클래스인 "기사" 클래스는 다음과 같다.

```
Abstract class Article {
    abstract void register( );
    abstract void remove( );
    abstract void modify( );
}
```

Concrete Item 클래스인 "FileArticle"은 다음과 같다.

```
class FileArticle {
    void register( ){....//파일로 된 글을 등록할 수
```

있는 행위 정의};
void remove( ){...}
void modify( ){...} }

4.4 활용방법

설계패턴들은 전문가들에 의해, 추출되어야 하고 그들은 특정 기준에 따라 체계적으로 분류돼야한다. [9] 분류된 설계패턴들은 OOIDM내에서 재사용가능한 네비게이션 설계, 인터페이스 설계, 실행가능 설계 단계에서 채택되어 구현될 수 있다. 개발자들이 패턴을 선택하고 구현하는 과정은 아래와 같다.

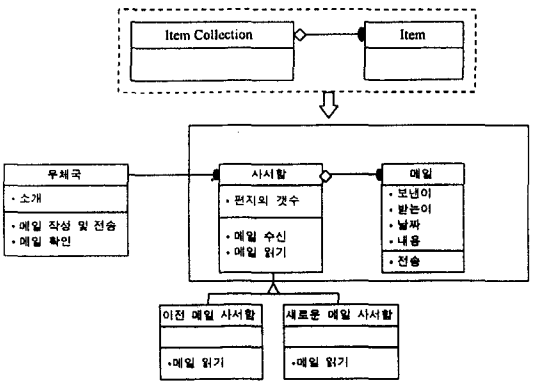
①먼저 추출된 패턴들이 어떻게 설계 문제를 해결하는지, 고려한다.

②패턴들의 목적 및 동기를 살펴본다.

③자신의 설계 문제가 패턴의 목적 및 동기에 부합되는지 대비시켜본다.

④일치한 경우, 설계패턴의 구조도와 구현사례를 중심으로 자신의 개발환경에 맞게 적용한다.

패턴의 선택 및 적용의 문제는 자동적으로 해결될 수 없는 문제이며, 인간의 경험이 크게 작용하게 된다. 설계패턴들은 여러 어플리케이션에 응용됨으로서, 재사용의 범위를 확장할 뿐 아니라, 이미 설계 정보가 결정되어 있기 때문에 개발시간을 단축시킬 수 있다. 위의 Container패턴의 예를 보면 "Item Collection"과 "Item"은 다양한 어플리케이션 영역에 적용할 수 있는데, 만약 전자메일 시스템의 경우엔, "사서함"(Item Collection)과 그 사서함에 배달되는 "메일"(Item)이라는 클래스로 (그림 8)과 같이 구현될 수도 있다. 이렇듯이 한 설계패턴은 설계이 유사한 여러 어플리케이션 영역에서 구현될 수 있다.



(그림 8) 메일 시스템에 적용된 Container 패턴
(Fig. 8) Container pattern applying for mail system

5. 결 론

본 논문에서는 인트라넷 영역의 어플리케이션을 개발할 때 사용될 수 있는 객체지향 개발방법인 OOIDM을 제안하였다. OOIDM을 통하여 어플리케이션을 개발할 경우에 다음과 같은 장점이 있다. 첫번째로 실세계를 쉽게 모델링할 수 있다. 이것은 객체지향 개발방법을 적용하여 실세계에 존재하는 객체를 그대로 어플리케이션에 대응함으로서 얻어진다. 두 번째는 시스템의 행위를 더욱 용이하게 모델링할 수 있다. 객체지향 시스템에서는 객체는 속성과 메쏘드로 이루어져 있는데, 이 메쏘드가 객체의 서비스이다. 시스템의 행위는 이런 객체들의 서비스의 집합으로 이루어지는데 본 방법에서는 객체의 서비스를 인트라넷 어플리케이션의 요소로 쉽게 대응될 수 있도록 하였다. 세 번째로는 인트라넷 영역에서 설계패턴을 사용하는 방법을 제시함으로서 재사용의 범위를 확장하였다. 설계패턴은 검증된 설계 정보이고 여러 객체들로 이루어져 있기 때문에 설계패턴을 재사용하는 것은 전체 개발시간의 단축과 유지보수성을 향상시키는 효과를 가져온다.

본 방법론은 특히 실세계를 그대로 인트라넷 어플리케이션으로 대응시켜서 구현할 경우와 규모가 크고 복잡할 경우에 적합하다. 향후연구과제로 다음과 같은 것들이 필요하다.

첫 번째로, 인트라넷 어플리케이션의 동적모델에 관한 연구이다. 현재 OOIDM은 동적모델은 다루고 있지 않고 있지만 인트라넷 어플리케이션이 사용자가 발생한 이벤트 위주 어플리케이션이라는 점에서 동적모델이 중요한 역할을 하기 때문에 필요한 부분이다. 두 번째로 설계패턴들의 조립을 통해 어플리케이션을 완성하기 위해서 패턴들의 대상 및 분류의 좀 더 공식적인 체계화가 필요하다. 세 번째로 설계패턴들의 분류, 검색, 획득, 적용을 지원하는 프레임워크 등에 관한 연구가 필요하다. 재사용과정을 자동화함으로써 재사용비용 및 시간을 단축할 수 있다[14].

참 고 문 헌

[1] Frank Buschmann and Regine Meunier, "A system of Patterns," pattern languages of programing language 2, 1995.

[2] Francois Fluckiger, "Understanding Networked Multimedia", prentice-hall, 1995.

[3] Gamma. E., Helm. R., Johnson. R., Vlissides. J., "Design Patterns: Elements of Reusable Object-Oriented Software," Addison-Wesley, 1995.

[4] Athula Ginige, David B. Lowe et al, "Hypermedia Authoring," IEEE Multimedia, Winter, 1995.

Wesley, 1994, p97.

[6] Tomas Isakowitz, Edward A. Strohr and P. Balasubrananian, "RMM: A Methodology for Structrued Hypermedia Design", Communication ACM 38, August, 1995, 34-44.

[7] Lange, D.B., "Object-oriented hypermodeling of hypermedia supported information systems," Proceedings of the 26th Hawaii International Conference on System Sciences, 380-389, 1993.

[8] Robert Orenstein, "A Pattern Language for an Essay-Based Web site," pattern languages of program design 2, Addison-wesley,1995.

[9] Wolfgang pree, "Design patterns for object oriented software development", addison-wesley, 1995.

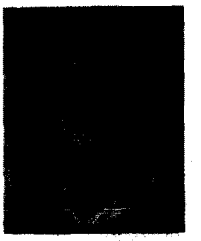
[10] Rumbaugh J., Blaha, M., Premeralni, W., Eddy, F. and Lorensen, W. "Object Oriented Modeling and Design." Prentice Hall, 1991.

[11] Dauglas C. Schmidt, "Experience Using Design Patterns to Develop Reusable Object-Oriented Communication Software", Communication of ACM, Octobjer 1995.

[12] Jiri Soukup, "Implementing Patterns, pattern languages of program design," Addison-wesley, 1994.

[13] 김종호, "뷰를 이용한 하이퍼미디어 어플리케이션 설계방법론", 석사학위논문, 한국 과학기술원 경영정보학과, 1995.

[14] 김정아, 이경환, "객체지향 소프트웨어 개발을 위한 재사용 지원 시스템", 정보과학회논문지(C), 제1권 제2호, 1995.



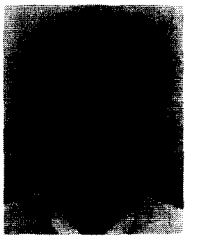
배 제 민

1991년 중앙대학교 전자계산학과(이학사)

1993년 중앙대학교 전자계산학과(이학석사)

1994년~현재 중앙대학교 컴퓨터공학과 박사과정

관심분야: 소프트웨어공학, 객체지향 방법론, 소프트웨어 재사용



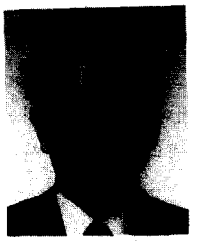
이 창 훈

1987년 광운대학교 전자계산학과(학사)

1989년 중앙대학교 전자계산학과(이학석사)

1994년~현재 중앙대학교 컴퓨터공학과 박사과정

관심분야: 소프트웨어공학, 멀티미디어시스템, Formal



이 경 환

1980년 중앙대학교 대학원 응용수학 전공(이학박사)

1982년~1983년 미국 Auburn대학 객원교수

1986년 서독 Bonn대학 객원교수

1971년~현재 중앙대학교 컴퓨터공학과 교수

관심분야: 소프트웨어 공학, 객체지향 모델링, 소프트웨어 재사용