

설계패턴을 이용한 캠퍼스 인트라넷 구조의 객체모델링

임 근*

Object Modeling of Campus Intranet Structure using Design Pattern

Keun Lim*

요 약

최근에 설계패턴 방법이 효율적이고 실용적인 재사용 기법으로 부각되고 있다. 설계패턴은 특정 도메인에 대한 여러 어플리케이션들 간의 공통된 행위들을 지닌 클래스들의 집합으로, 재사용성을 증가시키고, 어플리케이션 개발 시간을 단축시키는 효과를 가지고 있다. 그러나 설계패턴을 구축하기 위한 체계적인 개발 프로세스와 세부적인 개발 지침에 대한 연구가 충분히 이루어지지 않은 상태이다.

본 논문에서는 설계패턴을 기반으로 한 인트라넷 어플리케이션 구조의 모델링 기법을 제시한다. 제시된 모델링 기법의 프로세스는 영역분석, 네비게이션 설계, 인터페이스 설계, 행위영역 설계로 구성되어 있으며 각 단계는 논리적인 개발 업무들로 정의된다. 제시된 프로세스를 적용함으로써 보다 효율적인 인트라넷 모델링이 될 수 있다.

Abstract

Recently, design patterns have been known to be highly effective and practical. Design pattern is a collection of collaborating classes that captures the application independent behavior of system in a particular domain. By using design pattern, the development time and software quality can be greatly increased. However, systematic development process and detailed instructions of building design patterns have not been studied enough.

In this paper, we propose practical intranet structure modeling technique that are based on design pattern. The proposed process consists of the four typical software development phases: analysis, navigation, interface, behavior domain design, and each phase is defined as a logical sequence of development tasks. Since the proposed process is based on design pattern foundation, and can be efficiently produced by applying this process.

* 서울보건대학 전산정보처리과 조교수
논문접수: 1999.9.15. 심사완료: 1999.11.29

I. 서론

인트라넷 어플리케이션은 하이퍼미디어 형식을 통해 기존의 어플리케이션들을 통합하여 나타나기 때문에 단순한 정보획득에 목적을 두는 것이 아니라 정보의 전달, 수정, 삭제 등의 복잡한 양상을 보인다. 따라서 정보들 사이의 하이퍼미디어 연결관계를 이용한 정보획득이란 측면 뿐만 아니라 시스템의 행위적인 측면도 중시되어야 한다 [1]. 이러한 인트라넷 어플리케이션의 규모가 복잡하고 커지면 시스템의 행위로 인하여 비체계적인 방법으로 개발될 가능성이 증가되어 개발생산성과 유지보수성도 어려워질 것이다. 따라서 본 논문에서는 객체지향 방법을 기반으로 하고 적용대상으로 캠퍼스 정보를 모델링하는 캠퍼스 인트라넷 객체모델링 방법을 제안한다.

객체지향 개발방법론을 통해서 얻을 수 있는 이점들은 많지만 코딩 단계에서 생성된 부품들을 재사용 할 때는 그것을 자신의 개발 환경에 맞게 수정을 해서 사용해야 된다[2,3]. 이는 설계 단계에서의 변경이 발생하면 코드 자체에 많은 수정을 해야하며, 추가비용의 부담으로 인해서 재사용 효율을 저하시킬 수 있다. 따라서 코딩 전 단계인 설계 단계의 정보를 재사용 하는 개념이 설계패턴이다[4]. 설계패턴은 특정한 문맥 안에서 자주 발생하는 설계 문제들을 해결하기 위한 객체와 클래스간 통신 및 관계를 표현한 것이다. 본 논문은 복잡한 실세계를 구현, 유지보수시 효율적인 방법론인 객체지향 방법론을 이용하여 인트라넷 환경에 맞게 객체지향 방법을 적용하였다.

II. 관련연구

1. 기존 하이퍼미디어 어플리케이션 개발 방법론

기존의 하이퍼미디어 어플리케이션 방법론들은 크게 구조적 분석 방법론과 객체지향 개발 방법론으로 나누어진다. 구조적 분석방법론은 ERD[5]를 사용하고, 객체지향 개발방법론은 서비스를 수행하는 객체에 근거하여 모델링하는 방법이다.

1.1 RMM

ERD는 많은 시스템 분석가들에게 익숙하기 때문에 이를 근거한 분석은 쉽게 적용될 수 있다는 장점이 있다. 그러나 시스템의 행위를 결정하는 단계가 개발 단계 후반부에 독립적으로 존재하여 개발 초기에 영역을 분석할 때 시스템 행위를 모델링 하기 어렵다는 단점이 있다. 핵심적인 단계는 총 7개이며, 각각의 단계에는 가이드라인이 존재한다. RMM[5]은 주로 평범한 내부구조와 정보를 자주 갱신해야하는 하이퍼미디어 환경에 특히 적합하다.

1.2 OOHDM

이 방법은 개념적 설계, 네비게이션 설계, 추상적 인터페이스 설계, 구현의 단계로 이루어진다. 개발단계는 반복적이며 프로토타입 방식으로 수행된다[6].

이 모델은 하나의 네비게이션 모델로부터 여러 가지가 나타날 수 있다. 사용자로부터 이벤트를 받아들일 수 있는 인터페이스 클래스를 통하여 사용하는 네비게이션을 할 수 있다. 구현단계에서는 네비게이션, 추상적 인터페이스 모델을 구현 환경에 맞추어 구현하는 단계이다.

2. 설계패턴

설계패턴이란 이미 다른 전문가들이 작성해 놓은 코드나 설계의 패턴들을 추상화한 개념을 의미한다. 여러 가지 설계 패턴 접근 방법 중에서 특히 Gamma[3]등이 제안한 설계패턴 카탈로그의 개념이 객체지향 분야에 많은 영향을 주었다. 본 논문에서 설계패턴의 정의는 특정한 문맥 안에서 자주 발생하는 설계 문제들을 해결하기 위한

객체와 클래스들간의 통신 및 관계를 표현한 것이라고 할 수 있다[6,7].

본 논문에서는 기존의 객체지향 개발방법론에서 미약했던 기존의 개발지식을 활용할 수 있는 역할로서의 설계패턴을 활용한다.

III. 설계패턴 기반 객체모델링 기법

(Object-Oriented Modeling Technique based on Design Pattern : OMTDP)

본 논문은 기본적으로 OOADM의 라이프사이클을 인용한다. 다만 인트라넷 어플리케이션을 모델링 하기 위해 필요한 시스템의 행위를 추출하고 반영하도록 한다. 그림 1에서 시스템 행위를 구체적으로 설계하기 위한 행위영역 설계 단계를 추가하였다. 각 개발 단계를 통해 클래스들이 추출되며 최종 구현단계에는 인터페이스 클래스와 행위기능 클래스들이 구현된다.

1. 영역분석 단계

본 단계는 사용자로부터 요구조건문과 시나리오 기술서를 받아서 실제계에 존재하는 객체를 추출하는 과정이다. 이런 사용자 그룹별 시나리오를 통해서 시스템의 서비스를 추출할 수 있다[8,9].

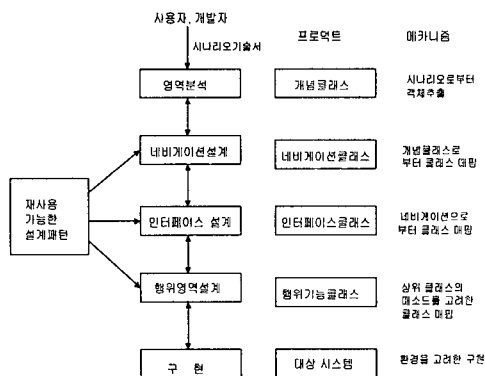


그림 1. OMTDP의 라이프사이클
Fig. 1 Lifecycle of OMTDP

이 같은 단계를 거쳐서 추출되는 클래스들은 개념 클

래스와 행위기능 클래스로 구분한다. 이 클래스의 속성은 사용자에게 보여지는 정보를 제공할 소스데이터이다. 행위기능 클래스는 사용자에게 직접 나타나지 않는, 클라이언트나 서버의 프로세서를 점유해서 서비스를 제공하는 클래스로서 나중에 행위기능 설계 단계에서 구체적인 내용이 결정된다.

2. 네비게이션 설계 단계

네비게이션 정보를 추출하기 위해 시나리오 기술서에 나타나 있는 시스템에 대한 요청과 획득에 대한 정보인 사용자 정보, 시스템 반응, 시스템 서비스를 참고한다. 예를 들어 수강시스템에서 교수와 학생이라는 개념 클래스가 추출되었고 사용자가 교수의 정보를 보고 있을 때 교수가 교수하는 학생의 정보를 보고자 할 경우 네비게이션을 해야 한다. 네비게이션 클래스내의 메소드의 형식은 method_id, anchor_name,

method_name, access_primitive, target_name 과 같다. method_id는 메소드의 순서로서 인터페이스 클래스에서 사용된다. anchor_name은 HC.앵커이름 또는 HC.앵커이름1, HC.앵커이름2, ... 등으로 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.대상이름1, HC.대상이름2,... 등으로 표기한다. access_primitive는 개발환경에 따라 추가할 수 있다. 그림 2는 기본접근방식에 대한 기호를 보이고 있다.

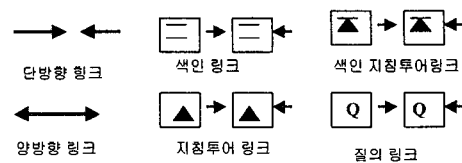


그림 2. 기본접근방식에 대한 기호들
Fig.2 Symbols about access primitives

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 되

고, 그 둘 사이에는 다대다(M : N) 관계가 성립한다. 만약 사용자가 특정교수의 정보를 HTML 형태로 보고 있을 때 해당교수가 저자인 출판물 정보를 획득하기 위한 접근 방식으로 교수가 저작한 여러 출판물을 가리키는 앵커를 통한 색인링크 방식을 선택한다. 이에 대한 하이퍼미디어 클래스와 네비게이션 클래스는 그림 3과 같다. 기호 N은 OMT에 추가된 표기법으로 두 개의 개념 클래스간의 네비게이션 정보를 갖고 있는 클래스를 나타낸다.

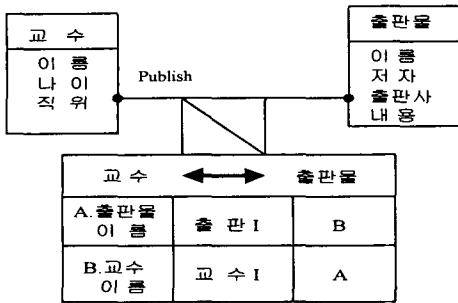


그림 3. 교수, 출판물에 대한 네비게이션 클래스
Fig. 3 Navigation class about Professor and publisher

3. 인터페이스 설계 단계

이 단계는 추출된 개념 클래스와 네비게이션 클래스를 기반으로 사용자가 인트라넷 어플리케이션을 직접 사용할 때 인터페이스 역할을 하는 인터페이스 클래스를 추출하는 단계이다. 그림 4는 개념 클래스가 네비게이션 클래스의 하이퍼링크 정보를 기반으로 인터페이스 클래스로 바뀌는 과정을 보이고 있다.

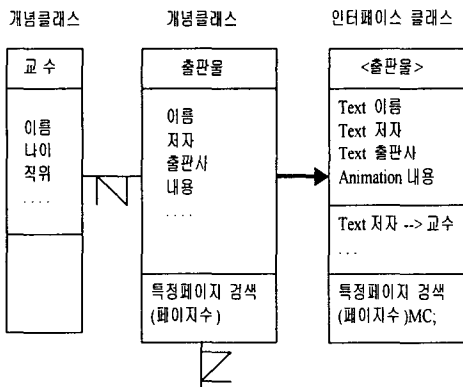


그림 4. 개념, 인터페이스 클래스 대응과정
Fig. 4 Mapping phase of concept and interface class

4. 행위영역 설계 단계

인트라넷 어플리케이션의 행위를 구체적으로 설계하는 단계로서 인터페이스 클래스와 네비게이션 클래스를 입력으로 받아서 인터페이스 클래스의 메소드와 속성 중에서 프로세스가 필요한 부분을 실제 시스템의 메모리와 프로세스를 점유할 수 있는 행위기능 클래스의 메소드로 대응한다. 이전 단계에 추출된 클래스간 관계는 다음과 같다. 즉 행위기능 클래스는 인트라넷 환경에서 CGI 프로그램이나 자바애플릿, 스크립트 언어와 같은 이동식 코드로 구현되게 된다. CGI와 MC가 객체지향 언어로 구현되어질 때 행위기능 클래스는 쉽게 구현될 수 있으나 객체지향 언어가 아닌 경우는 행위기능 클래스의 속성과 메소드를 가상적으로 구현해야 된다. 따라서 본 방법론으로 어플리케이션을 개발할 경우 CGI나 MC는 객체지향 언어로 구현하는 것이 바람직하다.

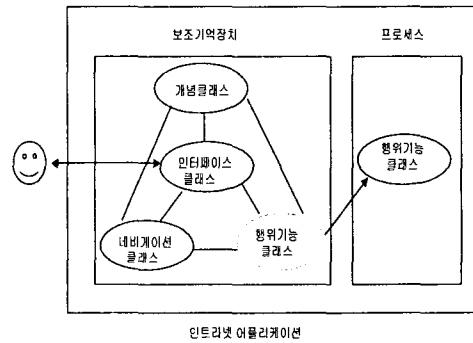


그림 5. 행위기능 클래스들간의 관계
Fig. 5 Relationship between behavior class and other classes

행위기능 클래스들의 메소드로 대응되는 인터페이스 클래스의 요소들은 인터페이스 클래스의 메소드 중에서 FORM-CGI, FORMLESS-CGI, MC, MC/APP와 인터페이스 클래스의 앵커 중에서 접근 방식이 질의 타입이거나 개념 클래스의 특정정보를 보여주기 위해 프로세스가 가공해서 사용자에게 공급할 필요가 있는 것, 그리고 인터페이스 클래스의 속성 중에서 텍스트스크롤, 애니메이션 동작 같은 특수효과가 필요한 부분은 MC화, 그리고 영역분석단계에서 추출된 클래스중 개념 클래스가 아닌 클래스는 본 단계에서 행위기능 클래스가 될 수 있다. 행위기능 클래스의 핵심서비스는 시나리오 기술서의 시스템 처리 부분을 고려하여 결정한다.

IV. OMTDP와 설계패턴

인트라넷 어플리케이션은 서버 쪽 구현과 클라이언트 쪽 구현으로 나눌 수 있다. 서버 쪽은 주로 CGI 프로그램이나 일반프로그램 언어로 작성된 어플리케이션이 동작하게 되고, 클라이언트 쪽은 HTML 및 MC, 즉 자바 애플릿이나 각종 스크립트언어가 실행되며 이런 것들이 서로 연합하여 하나의 완성된 어플리케이션을 이루게 된다.

1. 설계패턴의 대상

기존의 설계패턴은 시스템에 구현할 경우의 수단이 C 또는 C++ 같은 프로그래밍 언어로서 동적인 프로세스의 형태로 구현되었다. OMTDP에서 추출되는 클래스로 설계패턴을 표현할 때 다음과 같은 3부류로 대상을 나눌 수 있다

- o 하이퍼미디어 클래스와 네비게이션 클래스의 관계 및 상호동작
- o 개념 클래스, 네비게이션 클래스, 행위기능 클래스의 관계 및 상호동작
- o 행위기능 클래스들의 관계 및 상호동작

2. 설계패턴의 분류

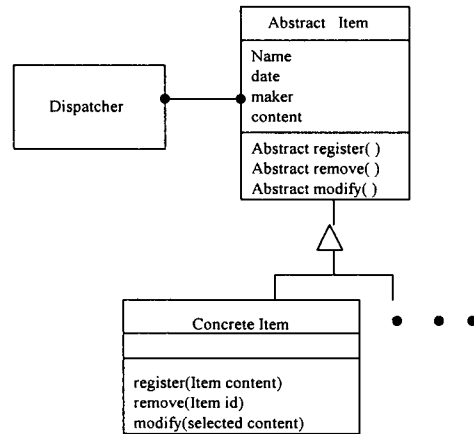
위의 3가지 단위로 표현되는 설계패턴들은 개발자가 자신의 개발환경에 맞는 설계패턴을 선택하여 이용하는데 도움을 주기 위하여 2가지 기준으로 분류할 수 있다 [10].

- o 추상화 정도는 어플리케이션 전체 아키텍처를 묘사하는 패턴으로 일반 패턴과 특수패턴으로 구분한다.
- o 구현 타입은 서버 패턴과 클라이언트 패턴으로 구분한다.

클라이언트의 경우 Item Collection 은 Item을 포함하는 클래스로서 Item을 관리하는 인터페이스를 제공하고 Item을 관리하는 메소드를 가지고 있다. Abstract Item은 추상 클래스로서 실질적으로 구현될 Item에 대한 인터페이스 역할을 한다. Concrete Item은 Abstract

Item으로부터 상속받은 클래스로서 구현되는 Item이고 이는 등록, 삭제, 수정할 수 있는 메소드를 가지고 있으며, 구체적인 행위를 정의한다.

* Client 구조



* Server 구조

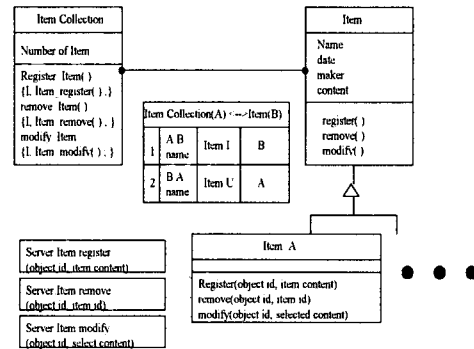


그림 5. 서버와 클라이언트 패턴구조
Fig. 5 Pattern Structure of Server and Client

3. 활용방법

설계패턴들은 전문가에 의해 추출되어야 하고 그것은 특정 기준에 의해 분류되어야 한다. 분류된 설계패턴은 OMTDP내에서 재사용 가능한 네비게이션 설계, 인터페이스 설계, 행위영역 설계 단계에서 채택되어 구현될 수 있다. 개발자들이 패턴을 선택하고 구현하는 과정은 아래와 같다.

- o 추출된 패턴들이 어떻게 설계 문제를 해결하는지 고

려한다.

- o 일치한 경우, 설계패턴의 구조도 와 구현사례를 중심으로 자신의 개발환경에 맞게 적용한다.
- o 패턴의 목적 및 동기를 살펴본다.
- o 자신의 설계 문제가 패턴의 목적 및 동기에 부합되는지 대비시킨다.

V. 결론

본 논문에서는 인터넷 영역의 어플리케이션을 개발할 때 사용할 수 있는 객체지향 개발방법으로 설계패턴 기반 객체모델링 방법(OMTDP)을 제안하였다. 이를 통하여 어플리케이션을 개발할 경우 다음과 같은 장점이 있다. 첫째, 실세계를 쉽게 모델링할 수 있다. 이것은 객체지향 개발방법을 적용하여 실세계에 존재하는 객체를 그대로 어플리케이션에 대응함으로써 얻어진다. 둘째, 시스템의 행위를 더욱 용이하게 모델링할 수 있다. 객체지향 시스템에서는 객체는 속성과 메소드로 이루어져 있는데 이 메소드가 객체의 서비스이다. 시스템의 행위는 이런 객체들의 서비스의 집합으로 이루어진다. 본 방법에서는 객체의 서비스를 인터넷 어플리케이션의 요소로 쉽게 대응될 수 있도록 하였다. 셋째, 인터넷 영역에서 설계패턴을 사용하는 방법을 제시함으로써 재사용의 범위를 확장하였다. 설계패턴은 검증된 설계 정보이고 여러 객체들로 이루어져 있기 때문에 설계패턴은 검증된 설계 정보이고 여러 객체들로 이루어져 있기 때문에 설계패턴을 재사용하는 것은 전체 개발 시간의 단축과 유지보수성을 향상시키는 효과를 가져온다. 본 방법론은 특히 실세계를 그대로 인터넷 어플리케이션으로 대응시켜서 구현할 경우와 규모가 크고 복잡할 경우에 적합하다. 향후 연구과제로 인터넷 어플리케이션이 사용자가 발생한 이벤트 위주 어플리케이션이라는 점에서 동적모델이 중요한 역할을 하기 때문이기에 대한 연구가 필요하며, 설계패턴들의 조립을 통해 어플리케이션을 완성하기 위해서 패턴들의 대상 및 분류의 체계적인 방법이 필요하다.

참고문헌

- [1] Frank Buschmann and Regine Meunier, "A system of Patterns", pattern language of programming language 2, 1995.
- [2] Francois Fluckiger, "Understanding Networked Multimedia", prentice-hall, 1995.
- [3] Gamma. E., Helm. R., Johnson. R., Vissides. J., "Design Patterns : Elements of Reusable Objected-Oriented Software," Addison-Wesley, 1995.
- [4] Athula Ginige, David B. Lowe et al, "Hypermedia Authoring," IEEE Multimedia, Winter, 1995.
- [5] Tomas Isakowitz, Edward A. Strohr and P. Balasubrananian, "RMM : A Methodology for Structured Hypermedia Design", Communication ACM 38, August, 1995, 34-44.
- [6] Robert Orenstein, "A Pattern Language for and Essay-Based Web site," pattern languages of program design 2, addison-wesley, 1995.
- [7] Wolfgang pree, "Design patterns for object oriented software development", addison-wesley, 1995.
- [8] Rumbaugh J., Blaha, M., Premeralni, W., Eddy, F. and Lorensen, W. "Objected Modeling and Design," Prentice Hall, 1991.
- [9] Dauglas C. Schmidt, "Experience Using Design Patterns to Develop Reusable Object-Oriented Communication Software", Communication of ACM, October 1995.
- [10] Jiri Soukup, "Implementing Patterns,

pattern language of program design.”
Addison-wesley, 1994.

저자 소개



임 근
1998년 2월 중앙대학교 대학원 컴
퓨터공학과 졸업
(공학박사)
1992년 - 현재 서울보건대학
전산정보처리과 조교수
관심분야 : 정보검색, 데이터마이
닝 제사용 방법