UML+Navigation Diagram 기반 웹 도메인 응용 개발 프로세스에 관한 연구

김 행 곤† · 신 호 준††

요 약

최근 웹 기술이 급속하게 발달함에 따라 웹 기반의 많은 애플리케이션이 개발되고 있다. 하지만 대부분의 웹 애플리케이션의 생성은 체계적인 프로세스 없이 개발자의 지식과 경험에 의존하여 개발되고 있다. 웹 기반의 응용시스템은 다양한 개발 기법이 요구되며 설계 및 개발 프로세스를 위해 풍부한 개발 방법론이 요구된다. 따라서, 본 논문에서는 추상화를 제공하지 않는 저수준 기술에 기반한 애플리케이션 개발을 개선하고 웹을 기반으로한 애플리케이션 구축을 위한 개발 프로세스를 제안한다. 또한, 웹 애플리케이션을 개발하기 위해서 분석, 설계 모델링 방법으로 네비게이션 다이어그램을 사용함으로써 체계적인 웹 애플리케이션 개발 프로세스를 제시하며, 전자 문제은행 시스템(EPBS : Electronic Problem Bank System)에 이 프로세스를 적용하였다. 본 논문에서 제시한 웹 애플리케이션 개발 프로세스는 모델링을 위한 고수준의 추상화 정의를 가능하게 함으로써 역으로의 개발을 통한 유지보수 정보획득으로 체계적 관리가 가능하며 모델 기반의 프로세스로써 이해하기 용이한 장점을 가진다. 또한, 분석과 설계 모델들은 이와 유사한 웹 애플리케이션 개발시 유용한 컴포넌트로써 재사용성을 기대할 수 있다.

A Study on the Process for Web Domain Applications Development Based on the UML+Navigation Diagram

Haeng-Kon Kim† · Ho-Jun Shin††

ABSTRACT

Recently, according to the rapid development of web technology, a lot of applications based on web techniques have been developed. However, most of web applications have been developed relying on knowledge and experiences of the developer without systematic process. Web Applications are seldom developed in isolation. For web application designers, the simple and semantically rich methodology is needed to improve design and development process. In this paper, we propose a new development process methodology to improve low level technology based application development process which do not provide high level abstraction. We also suggest a new methodology to construct applications based on web. We describe a systematic web application development process by using Navigation Diagram as a analysis, design modeling method to develop web application with productivity and quality. We apply the new development process to the EPBS(Electronic Problem Bank System) as examples. Web application development process proposed in this thesis can be maintained through reverse development, because it can be defined as high level abstraction for modeling. It is very easy to be understood as a process based on models. Also, analysis and design models can be reused as useful component whenever similar web application is developed.

† 종신회원 : 대구가톨릭대학교 컴퓨터정보통신공학부 교수
†† 준 회 원 : 대구가톨릭대학교 대학원 전산통계학과
논문접수 : 2000년 4월 4일, 심사완료 : 2000년 8월 23일

1. 서 론

인터넷의 확산과 더불어 웹이나 분산환경과 같은 새로운 소프트웨어 아키텍처를 기반으로 애플리케이션이 개발되고 있으며 그 대상 또한 기술적인 지식을 가지고 운영능력이 있는 개발자에서 비기술적인 사용자로 폭이 넓어지고 있다. 이러한, 소프트웨어 아키텍처와 새로운 사용자를 고려하여 멀티미디어, 그룹웨어, 전자상거래, 넷마케팅 등의 웹 애플리케이션이 생겨나게 되었다.

웹 애플리케이션은 많은 애플리케이션에 대한 아키텍처를 사용하는 것으로 이것의 장점은 이종의 클라이언트 공간, 소형의 클라이언트의 강력한 컴퓨팅 환경, 인터넷과 같은 분산된 네트워크에 잘 동작하는 능력을 포함한다. 오늘날의 웹 애플리케이션은 더 복잡해지고 관리하기 어려워지고 있다. 이러한 복잡성을 관리하는데 도움을 주기위해서 체계적인 웹 애플리케이션 개발이 필요하다. 그 하나의 방법으로 모델링은 중요한 부분을 차지하고 있고, 웹 애플리케이션에서 웹 페이지나 시스템 모델에 하이퍼링크와 같은 특정한 웹 구성요소들을 모델링하는 것이다. 또한, UML은 소프트웨어 집약 시스템을 모델링하기 위한 표준 모델링 언어지만[1], UML을 사용한 분석과 설계 모델은 페이지와 하이퍼링크를 모델링할 경우 몇몇의 문제가 발생한다. 박스 형태로 표현되는 UML은 웹 애플리케이션 모델링을 완전하게 만족시키지 못하며, 애플리케이션의 모든 타입에 대해 완벽하게 알맞지 않기 때문에 특별한 상황의 요구상황에 직면했을 경우 확장된 방법을 정의해야하는 것을 인정한다. 따라서, 웹 애플리케이션을 개발하기 위한 적절한 모델링 방법과 이를 이용한 분석과 설계에 관련된 프로세스에 관한 체계적인 연구가 요구된다.

본 논문에서는 소프트웨어 재사용을 목표로 웹 애플리케이션 개발을 위한 방법론으로 웹 환경을 지원하는 모델링을 사용하며, 인터넷/인트라넷을 고려한다. 분석과 설계 측면에서는 표준화된 모델링 언어인 UML과 웹 애플리케이션을 모델링하기위해 본 논문에서 제안된 Navigation Diagram을 사용한다.

2장에서는 본 논문에서 다루어질 영역인 웹 애플리케이션의 정의와 이와 관련된 웹 모델링과 기존의 개발 프로세스에 대해서 살펴보며, 3장에서는 웹 애플리케이션 개발 프로세스와 분석과 설계단계에서 모델링에 이용되는 Navigation Diagram의 정의와 UML과의 연관성을 기술하며, 4장에서는 사례 연구로써 Electronic Problem Bank System을 통해 제시된 방법론을 적용하며, 기존의 웹 애플리케이션과 비교, 평가한다. 끝으로 결론 및 향후 연구방향을 제시한다.

2. 관련 연구

2.1 웹 도메인 애플리케이션과 모델링

2.1.1 웹 애플리케이션

웹 애플리케이션은 일정한 비즈니스 로직을 가지고 웹 구현기술에 기반한 애플리케이션이라고 할 수 있다. 또한, 올바른 실행을 위해 웹에 의존하는 소프트웨어 애플리케이션이며, 웹 전반에 걸쳐 유도되기 위해 설계되어진다. 웹에 기반한 애플리케이션이나 웹 자체가 가지는 장점은 다음과 같다.

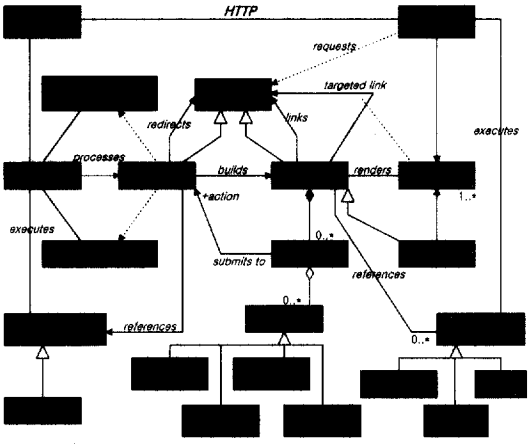
- 개발과 애플리케이션 사용의 용이
- 간단 명료하며 그래픽 사용자 인터페이스 제공
- 즉각적인 애플리케이션 분배
- 자동화된 다중 플랫폼 지원
- 다계층 클라이언트, 서버 아키텍처 지원

웹 애플리케이션은 설계모델에 초점을 맞추어 웹 구축자와 설계자를 대상으로 그 결과물을 웹 페이지로 한다. 하지만 모델링 작업의 이런 목적을 가진 웹 애플리케이션 설계는 명백하지는 않지만 모델링되어진 웹 페이지는 알기 쉬워야한다. 또한, 점점 복잡해지는 웹 환경을 관리하기 위해서는 모델링이 차지하는 부분은 아주 크다.

위의 (그림 1)은 일반적인 웹 애플리케이션 구조의 모델을 나타낸 것으로, 웹 서버와 클라이언트 브라우저 사이에 HTTP(Hyper Text Transfer Protocol)로 연결되어 있으며 웹 페이지를 중심으로 각각의 구성요소들 사이의 관련성을 나타내고 있다[2].

2.1.2 웹 구현기술

인터넷상에서 가용한 일련의 프로토콜이나 서비스를 제공하는 웹은 사용하는 측면을 고려할 때 브라우저, HTML(Hypertext Markup Language), HTTP 서버로



(그림 1) 웹 구조 모델

볼 수 있다. 각각은 웹 구현기술과 밀접한 관련이 있으며 웹을 기반으로하는 시스템이나 애플리케이션을 개발 할 때 꼭 염두해 두어야할 사항들이다. 또한, 웹 서버의 경우는 공개자료와 인터넷뿐만 아니라, 내부 정보의 분산을 쉽게 하고 다양한 정보의 공유를 위해 기업이나 기관에서는 인트라넷을 통한 직원 모집, 인사 및 정책안내, 안전규제, 영업정보에서 전자 결제시스템에 이르기까지 다양한 업무에 이용되고 있다. 현재는 웹 구현 기술을 통해서 웹 애플리케이션도 전형적인 애플리케이션과 마찬가지로 질의, 삽입, 갱신, 정보처리 등과 같은 사용자와 상호 동작하는 애플리케이션의 구현이 가능하다.

초기의 구현기술은 다른 HTTP 서버에 의한 다른 HTML 문서로 연결되는 참조 주소를 포함시켜 사용자가 문서상의 한 항목을 클릭함으로써 쉽게 다른 문서로 옮겨 갈 수 있는 크로스 참조 기능을 하는 전자 출판 모델이었다. 다음으로 등장하게되는 폼 기준 모델은 HTML 문서와 폼, 템플리트와 상호 연동할 수 있도록 CGI 프로세스를 사용하는 것이다. HTTP서버에 전달과 함께 즉각적으로 사용될 수 있는 장점으로 C나 C++ 뿐만 아니라 PERL(Practical Extraction and Report Language)과 같은 스크립트 언어로 작성되며, 일련의 기능을 수행하는 스크립트 라이브러리와 Add-on API (Applications Program Interface)들이 속속 나오고 있다. 최근에는 다계층 클라이언트, 서버 아키텍처로써 분산 처리를 위한 JavaBeans, Applet, ActiveX Control 등의 다양한 기술이 대두 되고있으며, 이에 대한 연구도 활발히 진행되고 있다.

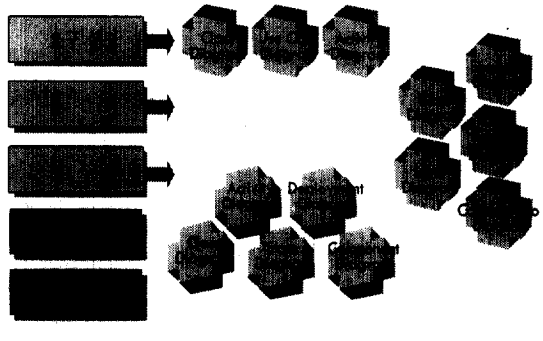
2.2 기존의 개발 프로세스

소프트웨어의 개발, 운영, 유지 보수, 그리고 폐기에 대한 체계적인 접근을 위해서 소프트웨어 공학은 허용하는 비용과 기간내에서 소프트웨어 제품을 체계적으로 생산하고 유지 보수하는데 관련된 기술적이면서 관리적인 원리에 입각해서 많은 방법론과 개발 프로세스들이 대두되었다.

2.2.1 UML 기반의 개발 프로세스

UML은 객체지향 시스템을 모델링하기위한 표준 다이어그램들과 표기법들을 규정하며 심볼들의 의미와 관련된 기본적인 의미론을 기술한다. 또한, 대규모의 복잡한 소프트웨어 시스템 개발에 필요한 가공물들에 대하여 모델링을 위한 구성 요소 제시, 이를 이용한 추상화 방법과 결과물 산출을 개발자들이 쉽게 이해할 수 있도록 가시화 하는 방법, 그리고 산출물 문서화 방법 등을 포괄적으로 정의한 모델링 언어이다[3, 4].

UML은 크게 4부분으로 구성되어 있다. 첫 번째가 사용자의 요구사항 정리에 필요한 Use Case모델링이며 두 번째가 현실세계의 복잡한 문제들을 추상화하여 풀어야 하는 문제의 범위를 개념적으로는 현실의 문제보다 더 광범위하게, 현실적으로는 현실의 문제보다 축소시키는데 필요한 객체, 클래스 모델링이다. 세 번째는 소프트웨어의 레고 블럭식 조립이 가능하도록 하는 개발 컴포넌트의 모델링이며, 네 번째는 개발 후의 소프트웨어 컴포넌트를 복잡한 분산처리나 클라이언트/서버 환경에 물리적으로 어떻게 배치할 것인지를 필요한 분산, 배치모델링이다.

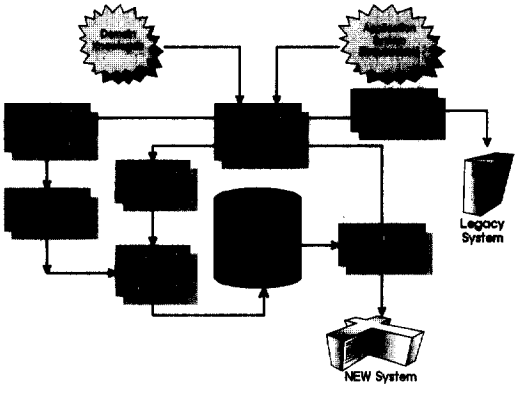


(그림 2) UML 기반 개발 프로세스

UML 개발 절차는 기존의 경험을 바탕으로 대규모의 복잡한 소프트웨어 시스템 개발을 위해 만들어진 것으로 다음(그림 2)과 같이 전체 프로세스를 나타낼 수 있으며 각 단계별로 모델링하기위한 다이어그램을 함께 나타내고있다[5, 6].

2.2.2 컴포넌트 기반의 개발 프로세스

잘 정의된 인터페이스를 통해서 사용되어진 의미있는 서비스를 유도하거나 기술되는 소프트웨어의 식별 가능한 조각인 컴포넌트(component)를 개발하기 위한 방법론은 최근에 주목받고 있는 부분이다. 컴포넌트 기반 개발은 OMG가 채택한 UML을 소프트웨어 표준 모델 도구로 모델 기반의 컴포넌트를 개발하기위한 단계와 이를 바탕으로 하나의 애플리케이션이나 시스템을 구축하는 과정이 병렬적으로 진행된다[2, 8]. 다음(그림 3)은 컴포넌트 기반 개발의 일반적인 프로세스로써 도메인 지식과 응용시스템 요구사항, 기존의 시스템에서 추출된 아키텍처를 기반으로 아키텍처 명세 및 검증과정을 거친다[7].



(그림 3) 컴포넌트 기반 프로세스

3. 웹 도메인 애플리케이션 개발 프로세스

3.1 Navigation Diagram과 Stereotype

기존의 웹 애플리케이션은 분석과 설계를 고려하지 않은 개발을 추진함으로써 본래의 목적을 수행하지 못하는 부적절한 파급효과(side effect)를 유발한다. 또한, 현재 웹 애플리케이션의 모델이 부족하고 모델링에 필요한 구성요소의 정의도 명확하지 않다. 따라서, 개발을 위해서는 체계적인 분석과 설계를 지원하는 모델링이 요구된다.

웹 도메인 애플리케이션의 개발을 위한 모델 기반의 프로세스를 지원하기 위해서 네비게이션되는 자료와 웹 페이지 등을 표현하는 방법으로 본 논문에서는 Navigation Diagram을 제시하고 분석, 설계에 도움을 주기 위해서 UML의 표준 확장메커니즘에 준하여 웹 페이지에 대한 stereotype을 정의한다.

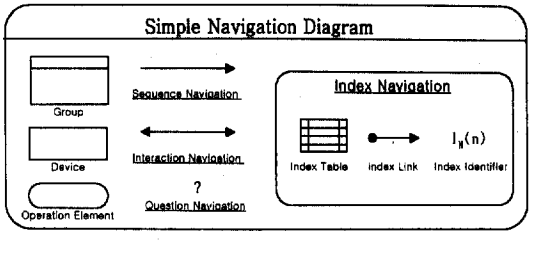
3.1.1 Navigation Diagram의 정의

모든 타입의 시스템 모델링에 적용가능한 UML은 표준화된 풍부한 표기법을 제공하며 실시간 시스템, 클라이언트/서버 그리고 많은 종류의 표준적인 애플리케이션 개발에 도움을 주고 있다. 하지만 통신망의 발전에 따라 많은 유용한 서비스가 제공되고 있으며, 자료 또한 정적인 이동에서 동적이고 다중적인 경로로 이동되고 있다. 인터넷/인트라넷에서의 네비게이션되는 정보의 흐름 순서는 하이퍼텍스트에서 문서를 열람하는 사용자에 의해 정해지는 것처럼 시스템 이벤트에 의해 정해진다. 이를 표현하기 위해 UML은 네비게이션되는 메시지가 장치사이의 흐름을 명세하는 표기법은 미흡하며 열악한 개발 프로세스를 가지고 있다.

UML 지식을 기반으로 동적이고 다중적인 경로를 통해 Navigation 되는 정보와 구성요소에 초점을 두고 웹 애플리케이션에 대한 분석과 설계를 위한 Navigation Diagram으로 표현되며, 기능적인 면과 모델되어지는 차이에 의해 Simple Navigation Diagram과 Use Navigation Diagram으로 나눈다.

1) Simple Navigation Diagram

인터넷/인트라넷 뿐만아니라 클라이언트/서버 환경에서 네비게이션되는 정보의 흐름과 연결에 관련된 시스템 전반적인 사항을 나타내기 위한 것이다. 전송되는 자료는 시스템 이벤트와 사용자 선택에 의해 정해지며, 이때 웹 구성요소사이에서의 다중적인 네비게이



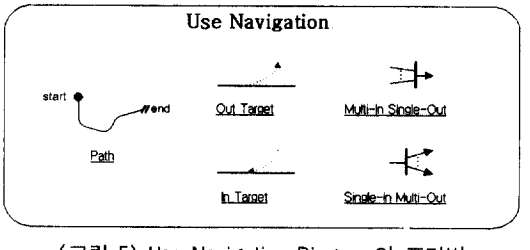
(그림 4) Simple Navigation Diagram 표기법

션 경로와 메시지를 표현한다. 네비게이션의 형태와 표기법은 위의 (그림 4)과 같다.

- Group : 하나이상의 장치나 오퍼레이션 요소를 하나의 영역으로 정의
- Device : 실제 자료들이 처리되고 메시지가 전달되는 각각의 구성요소로써 장비일 수도 있으며 일련의 작업을 행하는 프로세서
- Operation Element : 처리되는 자료의 행위를 표현한 것으로 UML의 Use Case와 유사한 개념
- Sequence Navigation : 네비게이션이 아직 표시되지 않았거나 단순한 순차적인 메시지일 경우
- Interaction Navigation : 경로를 설정하는 것과 같이 하나이상의 장치와의 상호관계를 나타낼 경우
- Question Navigation : 프로토콜의 설립과 같이 응답이 요구되는 질의가 필요할 때 사용
- Index Navigation : 장치의 응답이나 연결설정 등에서 볼 수 있는 다중 메시지를 나타낼 경우와 설정된 인덱스의 주소로 링크할 경우
  - I_N : N 번째로 구성된 인덱스의 번호를 정의
  - $I_N(n)$: N 번째로 구성된 인덱스의 n 번째에 해당하는 메시지나 처리

2) Use Navigation Diagram

전체 시스템에서 사용되는 오퍼레이션 구성요소의 사용을 나타내며 네비게이션되는 동안 전달되는 자료의 목적지를 명시하는 것을 원칙으로 한다. 이는 사용자에게 시스템 전반적인 구조와 실매개체를 직관적으로 파악하고 이해할 수 있는 장점을 가진다. 다음(그림 5)은 Use Navigation Diagram의 표기법을 나타내며 Simple Navigation Diagram과 연관해서 사용된다.



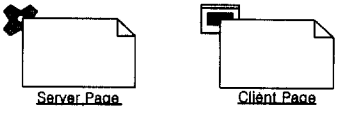
(그림 5) Use Navigation Diagram의 표기법

- Path : 구성요소를 네비게이션하는 동안의 모든 경로를 표현
- Out Target : 네비게이션되는 동안 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 것으로써 둘이상의 경로가 단일의 경로로 합쳐지며, 구성요소에서 유도될 수 있음을 표현
- SIMO(Single-In Multi-Out) : MISO와 유사하게 단일 구성요소에서 다중적인 경로로 뻗어나는 것을 표현

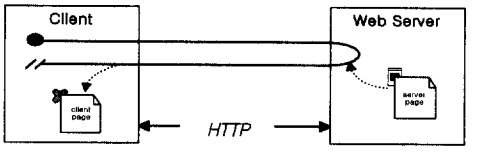
3.1.2 Stereotype 정의

웹 도메인의 지원 시스템은 다양하며 각각 많은 차이점을 가지므로 다르게 표현된다. 웹 애플리케이션은 설계모델에 초점을 맞추어 웹 구축자와 설계자를 대상으로 그 결과물을 웹 페이지로 하고 있다. 하지만 모델링 작업의 페이지를 객체로 표현하며 야기되는 문제를 고려해야만 한다. 즉, 객체의 속성과 구성요소를 표현하기 위해 사용하는 것이다.

설계모델과 웹 애플리케이션에서 사용되는 페이지 객체의 메소드로써 페이지에 포함된 스크립트를 식별하기는 어렵다. 특히, 웹 페이지는 클라이언트뿐만 아니라 서버에서도 스크립트를 포함할 수 있다. 따라서, 서버와 클라이언트 실행에 대한 메소드와 속성을 혼용하는 것은 매우 복잡한 일이다. 이 문제를 해결하기 위해 웹 페이지를 stereotype으로 정의할 필요가 있다. 다음(그림 6)은 stereotype으로써 설정을 아이콘으로 표현한 것이며[2], 본 논문에서 제안한 Navigation Diagram을 적용하여 일반적인 웹 애플리케이션 구조를 다음(그림 7)과 같이 주요한 구조적 컴포넌트를 중심으로 나타낼 수 있다.



(그림 6) 서버/클라이언트 페이지의 stereotype



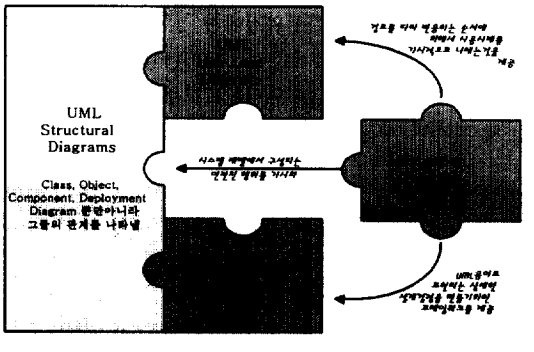
(그림 7) 기본 웹 애플리케이션 구조

3.1.3 UML과의 연관성

UML은 객체 지향 시스템을 모델링하기위한 표준 다이어그램들과 표기법을 규정하며 이들 다이어그램과 심볼들의 의미와 관련된 기본적인 의미론을 기술한다.

Use Navigation Diagram은 인터넷/인트라넷과 같은 도메인에서 전체 시스템에서의 사용을 나타내며 네비게이션되는 동안 주고받는 메시지를 도식적으로 나타낸다. 또한, 자료의 목적지를 명시하는 것을 원칙으로 함으로써 이는 사용자에게 시스템 전반적인 구조와 실매개체를 직관적으로 파악하고 이해할 수 있는 장점과 말로 표현된 명세와 상호작용 다이어그램에 의한 상세한 명세사이의 간격을 채울 수 있다. 다음(그림 8)은 UML과 Navigation Diagram의 연관관계를 도식화한 것이다.

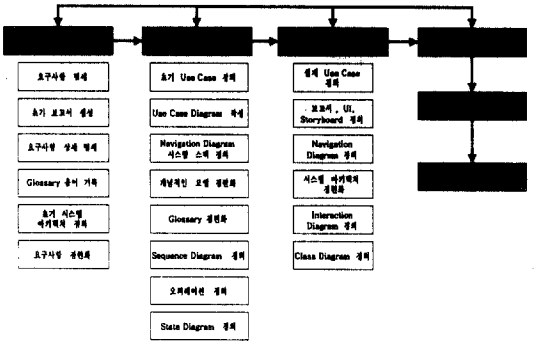
상호작용 다이어그램은 컴포넌트, 행위의 주체, 서브시스템, 객체들 사이의 상호작용에 관점을 두고 컴포넌트의 응답성은 주석으로 표현하게 된다. Navigation Diagram은 컴포넌트간의 메시지를 주고받는 것에 대한 상세한 표현이 없이 컴포넌트의 응답성에 대한 근거를 제공한다. Navigation Diagram을 사용하는 것은 상세한 설계를 할 때 동적인 행위를 멘탈모델(Metal Model)로 취할 수 있는 것으로부터 설계자는 자유롭게 될 것이다. 또한, Navigation Diagram을 사용사례와 객체 모델과 함께 반복적으로 되풀이함으로써 최고의 실행가능한 시스템 식별에 도움을 줄 것이다. 이는 더욱 구조화되고 견고한 재사용가능한 시스템을 구축할 수 있도록 한다.



(그림 8) UML과 Navigation Diagram과의 관계

3.2 개발 프로세스

웹 애플리케이션 개발 프로세스는 기존의 절차에 의존하여 개발하게 되며 웹 도메인의 요구사항과 분석, 설계 과정에 모델 기반으로 전개해 나간다. 다음(그림 9)은 웹 애플리케이션을 개발하기 위한 전체 프로세스와 UML과 Navigation Diagram을 이용한 분석, 설계 단계의 프로세스를 나타낸 것이다.



(그림 9) 전체 개발 프로세스

3.2.1 도메인 요구사항

개발 초기에 용역을 맡기는 고객의 요구를 충분히 알기 위해 그리고, 이러한 요구를 구축할 시스템에 충분히 반영하기 위하여 요구분석의 단계가 필요하다. 이 단계에서는 도메인에 속한 시스템, 소프트웨어, 사용자의 요구를 분석하기 위해서 초기 사용 사례에 대한 정의가 따르며, 향후 개발 시 사용자의 요구를 충분히 반영하기 위하여 문서화한다. 이때 작성되는 문서는 일반 텍스트 문서이며, UML에서의 출력은 Use Case Diagram이 될 수 있다. 또한, 문서화와 더불어 용어 기록은 중요한 정보가 되며 전체 단계에서 정련화되며 첨가 될 수 있다.

요구사항에 대한 전반적인 파악이 된 후에는 도메인에 대한 기본적인 지식을 바탕으로 초기시스템 아키텍처를 정의하고 요구사항의 정련화를 거치게 된다. 이 정보들은 도메인 분석 프로세스에 반영되며 개발이 완료 될 때까지 전체 프로세스에서 중요한 자료로 사용된다.

3.2.2 도메인 분석 프로세스

요구사항이 파악된 다음 실제 풀어야 할 문제를 분석하는 도메인 분석단계가 따른다. 하지만 세부적인 기술이나 특정 기술은 배제하고 웹 애플리케이션 도메

인 요소에 해당하는 모델을 고려한다. 도메인 분석 단계에서 먼저 행하여지는 일들은 도메인에 대한 지식을 얻어야 하며, 이는 기존 시스템의 기술, 사용자와의 대화, 일정한 비즈니스 로직, 용어의 분류 등이 될 수 있다. 이러한 지식은 초기 Use Case의 정의와 Use Case Diagram을 작성할 수 있으며, Navigation Diagram으로 시스템의 스펙을 정의하는 기반 개념이 될 수 있다.

Navigation Diagram으로 요구사항과 전반적인 소프트웨어 아키텍처를 설정하고 난 후 개념적인 모델로 정련화 한다. 또한, 요구사항에서 작성된 용어를 개념적으로 정련화하게 되며, 시스템의 일반적인 사항이 파악되면 웹 애플리케이션의 적당한 구성요소들의 후보들을 찾아야 한다. 이 과정에서 모든 후보들에 대해 필요 없는 구성요소들은 생략된다.

설정된 구성요소들 사이의 정적인 관계가 Sequence Diagram으로 모델링 되어진다. 또한, 구성요소 사이의 행위(behavior)와 협력(collaboration)등을 정의 함으로써 State Diagram으로 표현되어진다. 모든 다이어그램이 완성되면 종이에 시스템을 실행시켜보는 방법으로써 다시 한번 검증하는 것이 필요하다. 첨가적으로 기본적인 사용자 인터페이스의 프로토타입을 만드는 과정도 유용하다. 결과적으로 분석 단계에서 나오는 결과물로는 Use Case Diagram, Navigation Diagram, Sequence Diagram, State Diagram이 나오게 된다.

3.2.3 설계 프로세스

설계 단계에서는 분석 단계의 결과물에 기술적인 부분을 첨가하여 확장한다. 기술적인 확장이란 시스템을 어떻게 구현할 것인가에 초점을 두고 어떻게 동작하고 어떤 제약이 있어야 하는지에 관한 고려이다. 이와 같이 설계 단계와 기술적인 하부구조를 분리하는 것은 분석 단계에서 만들어진 결과를 되도록이면 변화시키지 않고 유지하면서 하부구조를 좀 더 쉽게 변화시키거나 발전시킬 수 있도록 하기 위함이다.

설계 단계에서 실제 일어나는 일은 분석 단계에 나온 구성요소들에서 기능적인 부분들을 분리시킨다. 설계를 위한 Use Case Diagram과 동시성을 가진 행위의 경우 공유되는 자원에 대하여 활동적인 구성요소와 비동기적 메시지, 동기화 기술을 가지고 Navigation Diagram이 모델링 되어야하며 기능적인 구성요소를 기반으로 시스템 아키텍처의 정련화 과정을 거친다. 마지막으로 Interaction Diagram과 설계의 주요 모델링 과정인 Class Diagram을 정의한다. 설계 단계에서의 결과물로 UML에서는 Use Case, Sequence, Collaboration, Class Diagram이며 Navigation Diagram과 보고서, 사용자 인터페이스 정의 등이 나오게 된다.

3.2.4 구현과 테스트

구현은 실제로 원시 코드를 작성하는 단계로써 이때 필요하다면 분석과 설계 단계를 반복한다. 이 단계에서는 다이어그램에서 특정언어의 구문으로 옮겨 적는 과정 그리고 필요하다면 컴파일하고 링킹하고 다시 디버깅하는 작업이 포함되어있다. 실제로 소스코드의 작성은 프로그래머에게 익숙한 작업이고 친근한 작업이지만 분석과 설계에서는 코딩을 먼저 하는 것이 바람직하지 못하며 분석과 설계의 과정을 충분히 거치지 않고 바로 코딩을 할 경우 분석하고 설계하는 단계를 다시 하는 경우가 빈번하며 이는 오히려 기존의 웹 애플리케이션과 마찬가지로 부적절한 파급효과나 개발을 장기화시킬 우려가 대단히 크다. 마지막 단계로써 테스트의 목적은 코드에서의 에러를 발견하고 분석과 설계 과정이 올바르게 진행되었는지에 대한 검증과 검사를 위한 단계다. 테스트 결과는 문서로 남게 되고 다음 버전에서 고쳐질 수 있는 근거가 되기 때문에 아주 중요하다.

4. 사례 연구

본 절에서는 전자 문제은행 시스템(EPBS : Electronic Problem Bank System)에 개발 프로세스를 적용하여 도메인 분석 모델링과 애플리케이션 개발 모델링의 핵심 다이어그램으로 학생관점에서 기술한다. 애플리케이션은 시험문제를 출제하는 교수와 문제를 풀게되는 학생을 고려한 시스템으로 다음(그림 10)은 개략적인 전체 시스템 구성도이다.

(그림 10) Electronic Problem Bank System 구성도

4.1 요구사항 명세

웹을 이용하여 성적 관리와 웹 애플리케이션 개발이

힘든 사용자 입장을 고려한 시스템으로, 전자 문제은행 시스템은 웹을 기반으로 교수가 주어진 입력폼에 문제를 출제하여 자동으로 문제 웹 페이지가 생성, 저장되는 과정과 학생이 문제를 풀고 성적을 확인하는 일련의 과정으로 이루어져있다. 다음(그림 11)은 사용자 요구사항을 간략하게 나타낸 것으로 데이터베이스와 PHP(Personal Home Page) 웹 구현기술을 사용하는 것을 고려한다.

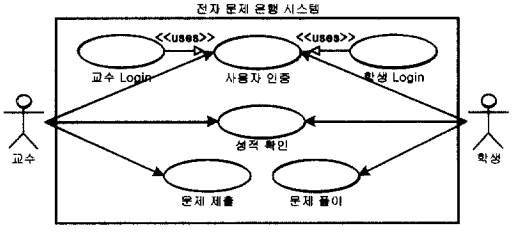
전자 문제은행 영역은 먼저 교수와 학생의 인증과정을 거치고 각각의 웹 페이지를 통해 서비스를 받게 된다. 관리는 교수에 의해서 하게 되고 교수페이지에서는 문제를 출제하게 되며, 출제된 문제는 서버에 저장되며 학생은 출제된 문제를 풀고 성적을 확인하는 서비스를 서버와 데이터베이스의 연동으로 처리하게 된다.

(그림 11) 요구사항 명세

4.2 도메인 분석 및 설계

4.2.1 Use Case Diagram

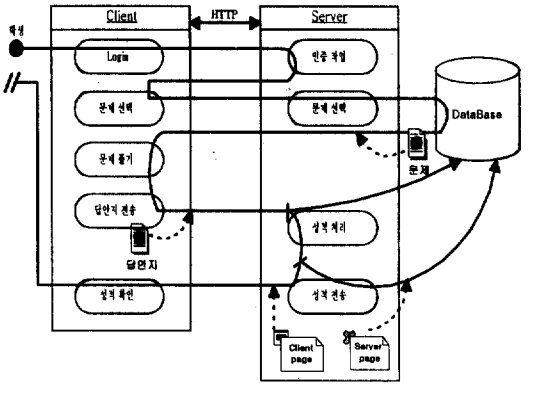
Electronic Problem Bank System 시스템의 학생과 교수의 Use Case Diagram은 다음(그림 12)과 같다. 웹 도메인에서 요구되는 사항은 사용자 인증, 성적확인, 문제 제출, 문제 출제 사용사례로 나타내며, 교수 Login과 학생 Login으로 사용자 인증 행위를 모두 상속한다.



(그림 12) Use Case Diagram

4.2.2 Use Navigation Diagram

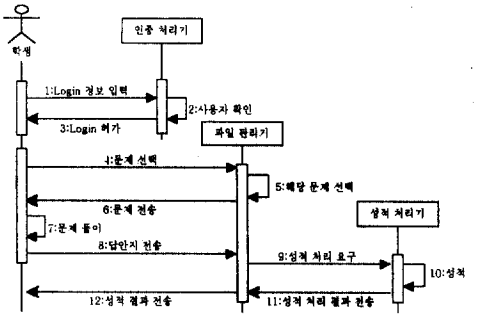
다음(그림 13)은 학생이 성적을 확인하는 과정을 Use Navigation Diagram으로 나타낸 것으로써, 서버 쪽의 서비스와 데이터베이스가 연동되는 것을 볼 수 있다. 또한, 성적을 처리하고 전송하는 부분에서는 데이터베이스에 저장과 학생에게 보여주는 페이지가 개념적으로 동일하지만 물리적으로 상이함으로써 앞에서 정의한 웹 페이지의 stereotype을 사용한다. 문제 선택과 문제 전송 오퍼레이션에서는 서버의 데이터베이스와 클라이언트 각각에서 In-Target되는 문제지 웹 페이지를 볼 수 있다.



(그림 13) Use Navigation Diagram

4.2.3 Sequence Diagram

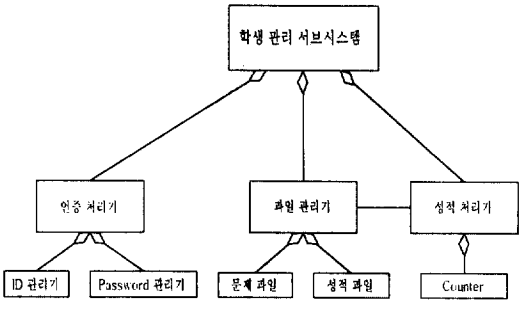
Use Navigation Diagram을 통해 분석된 내용은 Sequence Diagram에서 정적인 요소로 표현 가능하다. 다음(그림 14)는 행위자 학생이 인증 처리기, 파일 관리기, 성적처리기를 통해 처리되는 메시지를 시간의 흐름에 따라 도식화하고 있다.



(그림 14) Sequence Diagram

4.2.4 Class Diagram

설계 단계에서 모델링되는 클래스 다이어그램은 앞에서 모델링된 Use Case, Use Navigation, Sequence Diagram 등을 통해서 전체 시스템 중에 학생관리 서브 시스템을 다음(그림 15)과 같이 나타내었다. 구성은 인증 처리기, 파일 관리기, 성적처리기로 집단화(ag-



(그림 15) Class Diagram

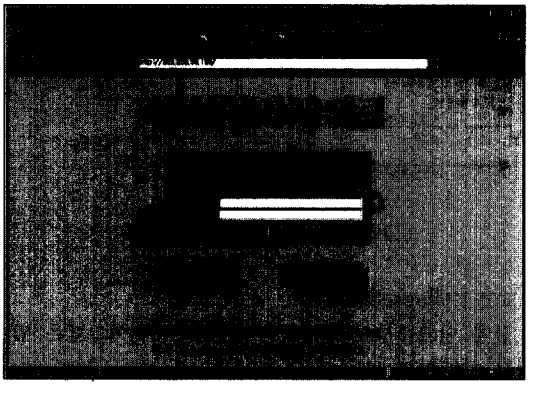
gregation)되어 있다. 또한, ID 관리기, password 관리기, 문제 파일, 성적 파일, counter들로 각각 집단화되어 있다.

4.3 구현 및 평가

UML과 Navigation Diagram을 통해 분석, 설계된 내용을 바탕으로 실제 웹 애플리케이션을 구현하는 단계로써 Linux 환경에서 서버에 PHP(Personal Home Page)를 이용하여 Electronic Problem Bank System을 구현하였다.

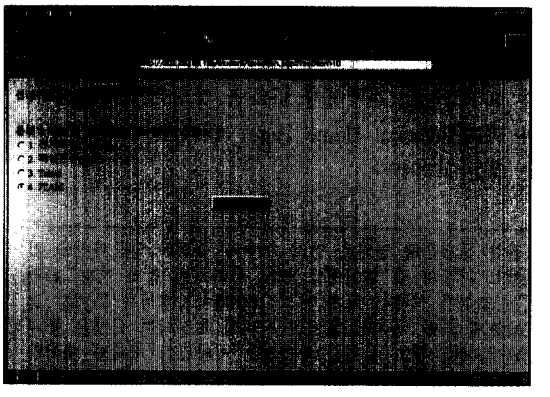
4.3.1 구 현

초기화면은 학생과 교수 영역으로 나눠지며, 선택된 영역에 대해서 다음(그림 16)과 같이 사용자 인증을 위한 창이 제시된다. 사용자 인증을 마치면 시험 출제를 위한 부분과 성적을 관리하는 부분이 나타나며, 시험 출제 영역에서 원하는 과목을 선택는 창이 제시된다.



(그림 16) 사용자 인증 화면

선택된 과목에서 실제로 문제를 출제하여 학생이 문제를 확인하고 풀이하는 과정에서 제시되는 창은 다음(그림 17)과 같다.



(그림 17) 학생 영역(문제 풀기)화면

4.3.2 평 가

본 절에서는 UML과 Navigation Diagram을 이용한 웹 애플리케이션의 개발 프로세스를 적용한 사례연구를 기반으로 기존의 애플리케이션을 비교 평가한다. 기존의 웹 애플리케이션은 개발 방법론 측면에서는 구현에 초점을 맞추고 분석과 설계에 대한 고려없이 웹 구현기술에 바탕을 두고 스크레취에 의한 개발을 주로 하고 있다. 또한, 분석, 설계를 위한 모델링 정보를 포함하지 않으며 프로세스에 대한 언급이 없다.

본 논문에서 제안하는 웹 애플리케이션 개발 프로세스는 모델에 기반한 절차로써 사용자와 소프트웨어, 시스템에 대한 요구사항을 고려하여 UML과 제시한 Navigation Diagram을 이용하여 분석, 설계 모델을 만든다. 또한, 체계적인 개발 프로세스를 통해서 웹 애플리케이션을 개발함으로써 기존의 웹 애플리케이션에 보다 더 유지보수성을 기대할 수 있다. 또한, 유사한 시스템 개발에 분석, 설계 모델은 재사용 가능하다.

5. 결 론

웹 기술을 이용한 애플리케이션의 개발은 저가의 구축비용과 개발의 용이성, 표준화된 문서환경, 다중 플랫폼 지원 등의 장점으로 웹 기반의 시스템을 구축하는 기업과 조직이 늘고있다. 하지만, 웹을 기반으로 개발되는 애플리케이션은 저수준의 구현기술을 가지고

있으며 적절한 개발 프로세스를 제공하는 것에는 아주 낮은 수준이다. 또한, 웹 애플리케이션이 요구사항의 명세나 도메인 분석, 설계를 고려하지 않은 임시적인 형태를 가짐으로써 부적절한 파급효과를 일으킬 수 있다.

본 논문에서는 구조적인 설계와 저수준 구현간의 격차를 줄이고 고수준 개념의 명세와 효율적인 유지보수성을 제공하기 위해 분석과 설계 모델링을 위한 UML과 Navigation Diagram을 이용한 웹 애플리케이션 개발 프로세스를 제시하였다. 또한, 웹 도메인과 관련된 애플리케이션으로 Electronic Problem Bank System을 사례를 들어 적용하였다. 결론적으로 다음과 같이 세 가지 측면으로 요약할 수 있다.

- **반복적인 프로세스**

기본적으로 기존의 개발 프로세스를 따르는 기법으로 사용자의 요구사항 분석, 설계 모델링과 아키텍처 정의를 UML과 Navigation Diagram을 사용하며 전체 프로세스가 안정적인 모델에 도달할 때까지 계속해서 반복하는 과정을 거친다.

- **모델 위주의 개발 기법**

프로그램 코드나 라이브러리 수준의 재사용이 아니라 모델 수준에서 재사용함으로써 유지 보수의 용이함을 획기적으로 기대할 수 있다. 최근 주목받고 있는 모델 기반의 컴포넌트 개발과 같은 맥락이라 볼 수 있다.

- **사용자 위주의 Navigation 모델링 기법**

웹 애플리케이션 개발을 위한 시스템의 요구사항을 현실 세계의 사용자 관점에서 바라보며 모델링하는 기법으로써 개발할 시스템의 전반적인 흐름을 보여주는 역할을 한다.

이 세 가지를 통한 웹 애플리케이션 개발과 관리는 사용자의 잦은 요구사항 변경이나 요구사항 추가, 삭제 등에 유연하고 탄력적으로 대처할 수 있으며 웹 애플리케이션 구축 후의 유지보수에 따르는 문제를 비교적 손쉽게 관리할 수 있다.

향후 연구방향으로는 웹 도메인 애플리케이션 개발의 표준 모델을 정하고 안정된 모델로 유도 할 수 있는 프로세스의 검증과 컴포넌트 기반의 시스템 개발 방법론을 통한 웹 애플리케이션 개발에 관한 연구가 요구된다.

참 고 문 헌

[1] Grady Booch, James Rumbaugh, Ivar Jacobson, *The Unified Modeling Language User Guide*, Addison-Wesley, 1998.

[2] Jim Conallen, "Modeling Web Application Design with UML," http : //www. rational.com/uml/resources/, 1998.

[3] Martin Fowler, Kendall Scott, *UML Distilled*, Addison-Wesley, 1997.

[4] OMG, "Unified Modeling Language Specification Version 1.3," http ://www.rational.com/, 1999.

[5] Craig Larman, *Applying UML and Patterns*, Prentice Hall, 1998.

[6] Ivar Jacobson, Grady Booch, James Rumbaugh, *The Unified software Development Process*, Addison-Wesley, 1998.

[7] Du-Hwan Bae, "CBSD : Component Based Software Development," Tutorial of The 11th KIPS Spring Conference, 1999.

[8] H. D. Hofman, "Componentware," Department of Mathmatics and Computing Cork Regional Technical College, 1997.

[9] Rational Software, "UML Extension for Objectory Process for Software Engineering version 1.1," http : //www.rational. com/uml/, 1997.

[10] Philippe Kruchten, "Modeling Component Systems with the Unified Modeling Language," 1998 International Workshop on Component-Based Software Engineering, 1998.

[11] Eun-Ju Han, "A Study on the Development of Framework Using Component Based Methodology", PD Thesis, Department of Computer Engineering, Catholic University of Taegu-Hyosung, 1999.

[12] Hans Erik Eriksson, Magnus Penker, *UML Toolkit*, Wiley Computer Publishing, 1998.

[13] Hans-W. Gellersen, Martin Gaedke, "Object-Oriented Web Application Development," IEEE Internet Computing, 1999.

[14] Jennifer Stone Gonzalez, *The 21st-Century Intranet*, PrenticeHall, 1998.

김 행 곤

e-mail : hangkon@cuth.cataegu.ac.kr

1985년 중앙대학교 전자계산학과 (공학사)

1987년 중앙대학교 대학원 전자계산학과(공학석사)

1991년 중앙대학교 대학원 전자계산학과(공학박사)

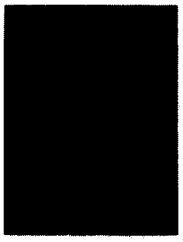
1978년~1979년 미 항공우주국 객원 연구원

1987년~1989년 한국전기통신공사 전임연구원

1988년~1989년 AT&T 객원 연구원

1990년~현재 대구가톨릭대학교 컴퓨터공학과 부교수

관심분야 : CBSE, 소프트웨어 재공학, CASE, 유지보수 자동화 툴, 요구공학 및 도메인 공학



신 호 준

e-mail : g98521002@cuth.cataegu.ac.kr

1998년 경일대학교 전자계산학과 (공학사)

2000년 대구효성가톨릭대학교 전산통계학과(이학석사)

2000년~현재 대구가톨릭대학교 전산통계학과 박사과정중

관심분야 : CBSE, 소프트웨어 재공학, CASE, 분석·설계 모델링, 도메인 공학