웹 어플리케이션의 효율적인 개발 환경 구축에 관한 연구

강 병 도† · 이 미 경††

요 약

인터넷의 급격한 성장과 함께 기존의 많은 소프트웨어들이 웹 기반으로 그 모습을 변화하고 있으며, 이로 인해 웹 어플리케이션의 복잡성이 증가되고 개발에 있어 많은 비용이 소요되고 있다. 따라서 웹의 특성을 잘 반영하는 개발 환경의 구축이 무엇보다 중요시된다. 본 논문에서는 웹 어플리케이션 개발을 위한 효율적인 환경을 제시한다. 이를 위해 웹의 특성을 파악한 후 웹 어플리케이션을 위한 프로세스와 모델링 환경을 정의한다. 제시된 환경은 크게 3가지 기능을 가진다. 첫째, 웹 어플리케이션 설계를 위한 모델링 환경을 제공한다. 둘째, WML이라는 웹 어플리케이션 모델링 언어를 제공한다. 셋째, 모델링 단계에서의 다이어그램을 바탕으로 자동으로 웹 페이지를 추출한다. 결국 제시된 환경의 사용은 웹 어플리케이션의 설계, 개발, 유지보수를 용이하게 할 수 있다.

A Study on an Efficient Environment for Web Applications Development

Byeongdo Kang† · Mi Kyong Lee††

ABSTRACT

Due to the rapid growth of Internet, modern software applications must support many web-based functionalities than traditional software applications. These web-based functional supports increase the complexity of software architecture and the cost of software development. Therefore, the development of an efficient environment that web characteristics are well reflected is the most important. In this thesis, we have presented an efficient environment for development of web applications. For the presented environment, after considering the web characteristics, we defined a process for web applications and modeling environment. The presented environment has three main functions : ① it provides a modeling environment for design of web-based applications, ② it has a modeling language called WML (web-application modeling language), ③ it automatically extracts web pages from diagrams. As a result, using the three main functions of the presented environment, we can easily design, develop, and maintain the web applications.

키워드 : 웹 공학(Web Engineering), 웹 어플리케이션 설계(Web Applications Design), 웹 어플리케이션 모델링(Web Applications Modeling)

1. 서 론

1960년대에는 하드웨어의 기술 발전속도가 소프트웨어의 기술 발전속도보다 빨라 새로운 소프트웨어를 요구하는 시장의 수요를 감당하기 어려워졌으며, 기존의 소프트웨어에 대한 유지보수가 힘들어졌다. 이러한 일련의 문제들은 인건비의 상승효과와 우수 소프트웨어의 부족현상으로 악화되어 소프트웨어 생산성이라는 새로운 과제를 해결해야만 하는 상황에 이르게 되었는데, 이를 흔히 소프트웨어 위기(software crisis)라고 한다. 이러한 현상은 소프트웨어 특성에 대한 이해의 부족과 관리의 부재 그리고 지나치게 프로그래밍에만 치중한 결과이다. 즉, 물리적이지 않고 논리적인 소프트웨어만이 가지고 있는 특성을 이해하지 못했고, 소프트웨어에 대한 잘못된 의식구조가 관리의 중요성을 소홀하게 만들었으며, 의사소통 장애와 효과적인 자원의 통제 등이 제대로 이루어지지 못했다. 또한 소프트웨어의 품질이나 유지보수성 등을 고려하지 않은 채 과거의 방식대로 프로그래밍만 잘하려는 집착으로는 복잡하고 다양해지는 소프트웨어의 요구수준을 맞출 수가 없었다. 따라서 이로 인한 개발 예산 초과, 개발 일정의 지연, 저조한 생산성, 미흡한 품질 등을 해결하기 위한 노력으로 소프트웨어 공학(software engineering)이라는 소프트웨어 개발에 있어 구조적이고 공학적인 접근방법을 시도하려는 노력들이 진행되었다[1]. 소프트웨어 공학에서 말하는 구조적이고 공학적인 접근법이란 소프트웨어의 개발을 단계적으로 추진하고

† 종신회원 : 대구대학교 정보통신공학부 교수
†† 준 회 원 : 대구대학교 대학원 컴퓨터정보공학과
논문접수 : 2002년 11월 27일, 심사완료 : 2003년 2월 7일

각 단계마다 표준방법을 채택함으로써 효율의 향상을 추구하고, 단계별로 미비한 점을 검토 및 보완시켜 나감으로써 품질을 보증할 수 있는 것을 말한다.

최근 네트워크의 발전 및 인터넷의 급격한 성장과 함께 기존의 많은 소프트웨어들이 웹 기반으로 그 모습을 변화하고 있다. 이로 인해 웹 어플리케이션의 규모와 복잡성 또한 크게 증가하고 있으며, 웹 어플리케이션 개발에 많은 인력과 비용이 소요되고 있다[2, 3]. 따라서 일반 소프트웨어에서와 같은 맥락으로 최근에는 웹 어플리케이션 개발을 위한 표준 방법론을 모색하고 생산성과 품질을 향상시키고자 하는 웹 공학(web engineering)에 대한 연구가 많이 진행되고 있다. 하지만 아직까지 웹 어플리케이션 개발을 위한 명확한 개발 단계의 정의나, 개발 단계별 역할 및 산출물 등에 대한 정의가 거의 없다. 대부분이 경험에 입각한 나름대로의 프로세스를 이용하여 개발하고 있는 실정이다[4].

소프트웨어를 개발함에 있어 분석과 설계는 그 프로젝트를 성공적으로 이끄는 중요한 요소이다. 분석은 사용자의 요구사항을 파악하는 단계로, 사용자와의 의사소통을 통해 시스템의 정보 흐름과 구조를 파악하고 기능의 세부적인 정의 및 인터페이스 설정 그리고 제약사항 등을 파악한다. 설계는 요구사항을 기준으로 소프트웨어 산물을 코딩할 수 있도록 표현하는 것을 말하며 개략 설계와 상세 설계로 나눌 수 있다. 개략 설계는 소프트웨어의 구조와 구성 요소들 간의 관계를 정의하고, 상세 설계는 구현 단위별 처리 절차를 정의한다. 따라서 이러한 분석과 설계단계에서의 결함은 소프트웨어 개발 전반에 치명적인 영향을 끼친다[1]. 지금까지 소프트웨어 개발에서 분석과 설계를 위한 다양한 기법들이 제시되어 왔다. 하지만 대부분 일반 어플리케이션을 위한 것이었다. 최근 UML의 확장과 같이 몇몇 웹 어플리케이션을 위한 기법들이 제시되고는 있으나, 웹 어플리케이션에 특성화되어 개발된 것이 아니므로 설계함에 있어서의 복잡성을 증가시켰다[2]. 일반적으로 웹 어플리케이션은 일정한 구조를 지니고 있고, 그 기능 또한 매우 제한적이다. 때문에 UML에서의 복잡한 다이어그램과 컴포넌트는 사용의 어려움을 가져왔다. 또한 웹 어플리케이션의 설계를 위해 기존 UML에서 정의된 컴포넌트의 확장이 이루어져야하므로 모델링 결과를 복잡하게 만든다.

따라서 본 논문에서 우리는 효율적으로 웹 어플리케이션을 개발하기 위한 환경을 구축하였다. 제시된 환경은 웹 어플리케이션의 설계와 구현과정을 통합한 것으로, 이는 크게 3가지 기능을 가진다. 첫째, 웹 어플리케이션 설계를 위한 모델링 환경을 제공한다. 이를 위해 웹 어플리케이션에 적합한 다이어그램과 컴포넌트를 새롭게 정의하였다. 이는 웹 어플리케이션의 정확한 분석과 설계를 도와주고, 분석과 설계에서의 결함을 줄여준다. 둘째, 다이어그램을 통한 모델링 결과를 저장할 수 있는 저장소로 WML(Web-application Modeling Language)을 제공한다. WML은 웹 어플리케이션의 기능과 구조적 특성을 기반으로 패턴을 정형화하고 구성 및 속성을 구체화한다. 이를 통해 모델링 결과의 저장이 가능하므로 기존 모델의 재사용을 가능하게 한다. 셋째, 모델링 결과를 바탕으로 실제 웹 어플리케이션 개발에 필요한 소스코드를 생성한다. 결국 소스코드 자동생성을 통해 웹 어플리케이션을 빠르게 개발할 수 있으므로 그 생산성을 높일 수 있다.

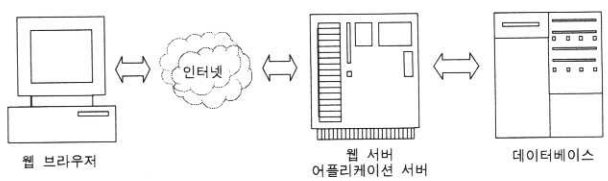
본 논문의 구성은 다음과 같다. 2장에서는 웹이 가지는 환경·기능적인 특성을 파악하고, 기존 웹 어플리케이션 개발 방법 및 기법들을 알아본다. 3장에서는 제시된 웹 어플리케이션 개발 환경의 구조와 기능을 설명하고, 4장에서는 구축한 환경을 실제 웹 어플리케이션 개발에 적용시켜 본다. 그리고 5장에서 시스템의 평가 및 고찰을, 6장에서 본 논문의 결론 및 향후 연구방향을 모색한다.

2. 관련 연구

2.1 웹 공학

웹 공학은 소프트웨어의 개발 예산 초과, 개발 일정의 지연, 저조한 생산성, 미흡한 품질 등과 같은 문제점을 해결하기 위하여 표준 방법론을 모색하고, 생산성과 품질을 향상시키기 위해 소프트웨어 공학을 웹 어플리케이션 개발에 적용시키려고 하는 것이다. 물론 소프트웨어 공학을 그대로 웹 어플리케이션 개발에 적용할 수는 없지만, 현재 웹 어플리케이션 개발에 있어서 필요한 표준 개발 방법론이나 프로젝트 관리 등을 소프트웨어 공학에서 이용할 수 있다.

초기의 웹은 텍스트 정보만 표시하는 간단한 웹 페이지들로 구성되었고 변화하지 않는 정적인 내용들이었다. 또한 단순한 페이지 간 링크 구조를 가지면서 외부 시스템과의 접속이 없는 독립형으로 존재했다. 따라서 소규모의 팀이 개발에 참여했으며 적절한 성능의 유지는 무시되었다. 그러나 웹의 기술이 점차 발전하고 사용자의 요구가 증가됨에 따라 웹사이트와 사용자간의 양방향 커뮤니케이션이 이루어지게 되었다. 즉, 웹 사이트도 동적으로 변화하게 되었으며, 데이터베이스와의 연동으로 시간 혹은 사용자에 의해 변화하는 정보를 가지게 되었다. 따라서 웹 어플리케이션 개발에 다양한 영역의 전문가들이 동원되는 대규모의 프로젝트팀이 요구되었다. 일반적으로 웹 어플리케이션은 (그림 1)과 같이 3-Tier 구조를 가진다.



(그림 1) 웹 어플리케이션 구조

오늘날 웹은 신문, 잡지, 책 등의 온라인 출판뿐만 아니라, 전자상거래, 금융, 마케팅, 광고에 이르기까지 다양한 분야에 걸쳐 우리의 삶에 영향을 미치고 있다. 기업에서도 웹은 제품 홍보, 판매, 마케팅 등 매출에 막대한 영향을 미치는 중요한 위치를 차지하고 있다. 또한 웹은 개인이나 기업, 국가에게도 그 중요성이 날로 증가하고 있고, 그 규모는 점차 커지고 복잡해져가고 있다[3]. 이와 같이 웹 어플리케이션의 복잡성이 증가함에 따라 컨텐츠에 대한 비용이 증가하고, 전자상거래 작업이나 빠른 업데이트 등 웹 사이트에 대한 요구도 같이 증가하고 있다. 웹 사이트의 규모가 커지고 복잡해져감에 따라 소수의 웹 디자이너들이 웹사이트를 개발했던 것에서 개발에 필요한 인력과 그 역할을 나누게 되었고, 다른 제품처럼 납기를 위한 일정 관리가 필요하게 되었다. 또한 웹 사이트에 대한 시장의 요구가 많아짐에 따라서 생산성에 대한 문제가 발생했고, 그래픽 디자인, 정보설계, 탐색, 편집기술, 컨텐츠 등의 사용성을 향상시키는 등 품질 향상을 위한 노력을 하게 되었다.

하지만 지금까지는 웹 사이트 개발시 각 개발 단계에 대한 명확한 정의나 개발 단계별 역할 및 산출물 등에 대한 정의가 거의 없는 것이 사실이다. 대부분 경험에 입각한 나름대로의 개발 프로세스들을 가지고 있다. 웹 사이트의 실패는 마케팅이나 그래픽 디자인, 탐색, 정보 디자인 자체의 잘못도 있지만, 그 보다는 분석과 기획, 설계 단계의 결함이 더 심각하다. 테스트의 경우에도 단위 테스트, 통합 테스트, 사용환경별 테스트 등의 정형화된 테스트 방법론도 거의 없어서 다양한 웹의 사용자들에게 웹 사이트가 원래 원하지 않는 모습을 보여 줄 수도 있는 것이다.

요즘은 이미 개발된 웹 사이트에 대해서 BPR(Business Process Reengineering) 하듯이 다시 기획부터 시작해서 설계, 개발을 하거나 웹 사이트 평가나 교정에 대한 요구가 늘어나고 있다. 그러나 개발 단계별 산출물에 대한 정의도 없고, 개발 문서도 거의 없어서 나중에 유지보수나 웹 사이트 재구축 등을 할 때 심각한 문제가 발생하기도 한다. 국내에서도 많은 분야에서 품질에 대한 높은 관심으로 ISO 인증을 획득하기 위한 많은 노력을 하고 있다. 소프트웨어에도 여러 개의 ISO 인증 제도가 있다. 그러나 웹 사이트 개발에 있어서는 아직 품질보증 활동이 거의 없다. 너무도 급격하게 발전하고 변화해서이기도 하지만, 소프트웨어와 온라인 출판 등 어느 분야에도 확실히 속하지 않는 명확하지 않은 영역 때문일 수도 있을 것이다. 그렇지만 앞으로 웹 사이트에 대한 품질의 중요성이 커짐에 따라서 품질활동은 분명 그 중요성이 커질 것이다[5].

그 동안 웹 어플리케이션 개발에 있어서 일련의 과정들에 대한 정형화되고 표준화된 작업 방법이 없었던 것이 사실이다. 따라서 예산과 개발 일정을 고려하고, 더 나은 생산성과 품질을 얻기 위해 기획, 설계, 개발, 그래픽 디자인, 정보 설계, 컨텐츠, 테스트, 유지보수 등의 일련의 과정들을 프로세스화 하고, 구조적으로 접근하는 방법 등 웹 어플리케이션 개발에 있어서 소프트웨어 공학의 이러한 표준 개발 방법론이나 프로젝트 관리, 품질 관리 등을 적용시킨다면 현재의 웹 어플리케이션 개발에 있어 좋은 지침이 될 수 있을 것이다.

2.2 웹 어플리케이션 개발 방법 및 기법

최근 기존의 많은 소프트웨어들이 웹 기반으로 그 모습을 변화하고 있다. 이로 인해 웹 어플리케이션의 규모와 복잡성 또한 크게 증가하고 있으며, 웹 어플리케이션 개발에 많은 인력과 비용이 소요되고 있다. 따라서 최근에는 웹 어플리케이션 개발을 위한 표준 방법론을 모색하고 생산성과 품질을 향상시키고자 하는 웹 공학에 대한 연구가 많이 진행되고 있다. 이에 따라 지금까지 웹 어플리케이션 개발을 위한 여러 가지 개발 방법들이 제시되었다.

2.2.1 HDM(Hypermedia Design Model)

이태리의 Franca Garzotto교수에 의해 ACM에 발표된 방법론으로 하이퍼미디어 설계에서 시스템적, 구조적 접근 방법론으로, OOHDM에 많은 영향을 주었으며, 근래에는 하이퍼미디어의 유용성(Usability) 평가에 대해 깊은 연구 결과를 발표하고 있다[6].

2.2.2 RMM(Relationship Management Method)

RMM[7]은 뉴욕대학 정보시스템학과 교수인 Tom Isakowitz에 의해 발표되었다. 이는 웹 기반 정보시스템(WIS : Web-based Information Systems) 설계 및 개발자 지침을 제공하며, 관리의 효율성을 제공할 수 있는 방법이다. 즉, 웹 어플리케이션의 설계, 구축 및 관리를 위한 방법론으로 동적 DB를 유지하는데 소요되는 비용을 감소하고자 하는 데 목적이 있다. RMM은 웹 어플리케이션의 분석보다는 DB를 지원하는 동적 웹사이트를 생성하고 유지하기 위한 넓은 범위의 접근법이다.

2.2.3 OOHDM(Object-Oriented Hypermedia Design Model)

1995년 Rossi의 박사학위 논문으로부터 시작된 OOHDM[8]은 웹 어플리케이션을 개발하기 위한 방법으로 개념 설계, 네비게이션 설계, 추상 인터페이스 설계, 구현의 4개의 프로세스를 정의하여 복잡한 정보들을 간결하게 기술할 수 있도록 해주는 객체지향 프레임워크이다.

여기서 소개된 개발 방법론을 바탕으로 지금까지 다양한 웹 어플리케이션 개발 기법이 제시되었으나, 최근 가장 많이 사용되는 기법은 확장된 UML을 이용하는 것이다. UML[9]은 객체 관련 표준화기구인 OMG에서 1997년 11월 OMT (object modeling technique), OOSE 방법론 등을 연합하여

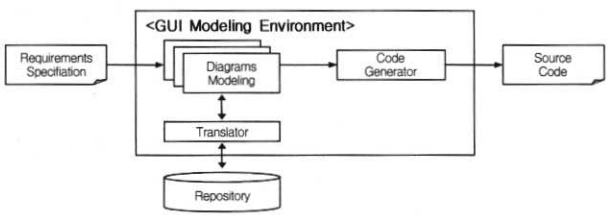
만든 통합 모델링 언어이다. 이는 요구 분석, 시스템 설계, 시스템 구현 등의 과정에서 생길 수 있는 개발자간 의사 소통의 불일치를 해소할 수 있고, 모델링에 대한 표현력이 강하고 비교적 모순이 적은 논리적인 표기법(notation)을 가진 언어라는 장점이 있다. 이러한 UML의 확장을 통해 웹 어플리케이션을 모델링하는 방법은 W2000[10], JESSICA[11] 등 다양한 연구 결과를 가지고 있다.

하지만 웹 어플리케이션은 일반적으로 일정한 구조를 가지며, 사용되는 기능 역시 매우 제한적이다. 또한 웹 페이지들 사이의 항해를 한다는 점에서 일반 어플리케이션과 크게 구분된다. 이러한 점에서 기존 UML의 확장을 통한 웹 어플리케이션의 모델링은 사용의 어려움 및 모델링 결과의 복잡성을 가져온다. 즉, UML에서 사용되는 많은 다이어그램과 컴포넌트를 습득하고 사용함에 있어 어려움이 있고, 웹에 특성화되어 개발된 컴포넌트가 아니므로 웹 어플리케이션을 설계하기 위해 기존 UML 컴포넌트를 확장해야 하기 때문에 모델링 결과를 복잡하게 만든다. 따라서 웹에 적합한 모델링 환경의 필요성이 대두되었고, 이를 위해 본 논문에서는 웹 어플리케이션에 특성화된 모델링 환경을 제시한다.

3. 웹 어플리케이션 개발 환경

3.1 구 조

본 논문에서는 웹 어플리케이션을 효율적으로 개발하기 위하여 설계에서 구현까지의 과정을 통합한 개발 환경을 제시한다. 즉, 설계가 곧 구현으로 연결되므로 효율적으로 웹 어플리케이션을 개발할 수 있다. 또한 이 환경은 웹 어플리케이션에 특성화되어 개발되었다. (그림 2)는 본 논문에서 제시하는 웹 어플리케이션 개발 환경의 전체적인 구조이고, 그 기능에 따라 Modeling, Translator, Code Generator의 세 부분으로 나누어진다.



(그림 2) 웹 어플리케이션 개발 환경 구성도

3.1.1 Modeling

모델링 부분에서는 요구사항 명세서를 바탕으로 개발하고자 하는 웹 어플리케이션을 다이어그램과 그에 따른 표기법을 이용하여 설계한다. 모델링에 사용되는 다이어그램에는 Architecture Design Diagram, Navigation Design Diagram, Page Detail Design Diagram이 있고, 이들은 나름대로의 특성을 지니고 있다. Architecture Design Diagram은 개발하고자 하는 웹 어플리케이션의 기능을 구조화하여 트리 형태로 나타내고, Navigation Design Diagram은 웹 페이지들 사이의 항해(navigation) 관계를 표현하며, Page Detail Design Diagram은 웹 어플리케이션을 구성하는 각 페이지들의 상세 설계를 나타낸다. 하나의 프로젝트는 하나의 Architecture Design Diagram과 Navigation Design Diagram 그리고 여러 개의 Page Detail Design Diagram을 가지게 되며, 각 다이어그램의 작성 순서는 나열된 다이어그램의 순서를 따른다. 다이어그램을 구성하는 표기법은 크게 컴포넌트(components)와 커넥터(connectors)로 나누어지고, 그 사용은 다이어그램에 의해 제한된다. 또한 컴포넌트와 커넥터 사이의 관계 및 그들의 속성은 모델링 단계에서 결정되어 진다. 다이어그램과 컴포넌트 및 커넥터를 통한 모델링 환경의 제공은 웹 어플리케이션을 개발하는 입장에서 보다 시각적이고 구체적으로 시스템을 분석하고 설계할 수 있게 도와주며, 설계된 내용의 수정도 용이하게 한다.

3.1.2 Translator

Translator는 모델링 부분에서 다이어그램과 컴포넌트 및 커넥터를 이용하여 설계된 웹 어플리케이션을 모델링 언어로 변환하는 작업을 수행한다. 또한 모델링 언어는 다시 다이어그램의 형태로 변환이 가능하다. 모델링 언어는 시스템의 기능과 구조적 특성을 기반으로 하여 패턴을 정형화하고 구성 및 속성을 구체화한 것으로 웹 어플리케이션 모델링 결과를 저장하기 위한 저장소(repository)이다.

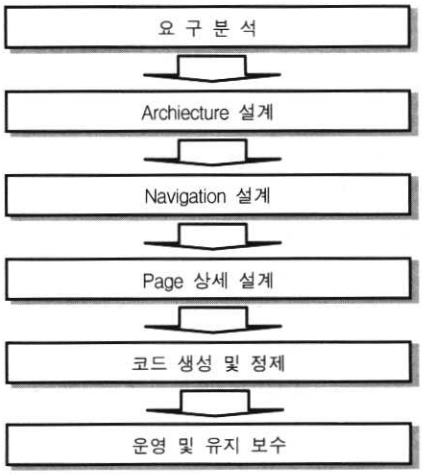
따라서 본 논문에서는 WML(web application modeling language)이라는 웹 어플리케이션 모델링 언어를 새롭게 개발했으며, 어플리케이션 모델링 결과를 WML의 형태로 저장한다. 이러한 저장소를 개발함으로써 기존 사용된 모델의 재사용을 높일 수 있다.

3.1.3 Code Generator

Code Generator는 모델링 부분에서 설계된 어플리케이션 모델링 결과를 바탕으로 실제 웹 어플리케이션을 구성하는 웹 페이지를 자동으로 생성한다. 이러한 웹 페이지들은 각 다이어그램을 구성하는 컴포넌트 및 커넥터의 관계 및 속성을 바탕으로 생성된다. 하지만 이렇게 생성된 소스코드는 완벽하지는 않으므로 시스템 개발자에 의한 정제 과정을 거치게 된다. Code Generator를 통한 자동 소스코드의 생성은 웹 어플리케이션의 빠른 개발을 가능하게 한다.

3.2 프로세스

본 논문에서 정의한 웹 어플리케이션 개발 프로세스는 (그림 3)과 같이 6단계로 나누어진다. 정의된 프로세스는 모든 웹 어플리케이션을 개발함에 있어 일관되게 적용되며, 각 단계별 수행하는 작업 및 산출물은 다르게 나타난다.



(그림 3) 웹 어플리케이션 개발 프로세스 모델

3.2.1 요구분석

요구분석 단계에서는 사용자가 원하는 시스템 요구사항을 정확히 파악하여 개발하고자 하는 웹 어플리케이션의 목표와 기능을 결정한다. 분석의 근본적인 목적은 사용자가 요구하는 문제에 대해 개발자 측면에서 정확하게 파악하여 각각의 문제에 대해 이해하기 쉽고 원활히 접근하고자 하는데 있다. 따라서 요구분석 단계에서는 사용자와의 의사소통 기술이 무엇보다도 요구되며, 사용자의 요구사항을 얼마만큼 이해했는가에 따라 그 프로젝트의 성공과 실패를 결정하게 되는 중요한 단계이다.

3.2.2 Architecture 설계

Architecture 설계 단계에서는 요구사항 명세서를 바탕으로 개발하고자 하는 시스템이 가지는 기능들의 전체적인 구조를 결정한다. 즉, 어플리케이션을 기능별로 구분하여 각 기능에 따른 상하·수평관계를 결정하여 트리 형태로 표현한다. 잘 정의된 구조는 소프트웨어를 개발함에 있어 그 시스템의 복잡성을 줄여주고, 대규모 프로젝트의 경우 각 팀별로 작업한 내용을 통합하기 쉽게 도와준다. 이 단계에서 발생되는 산출물은 Architecture Design Diagram 이다.

3.2.3 Navigation 설계

Navigation 설계 단계에서는 웹 어플리케이션을 구성하는 웹 페이지들 사이의 항해(navigation) 관계를 정의한다. 즉, 웹 페이지들 사이의 연결 관계 및 데이터 이동을 정의한다. 웹 어플리케이션은 웹 페이지들로 이루어지고, 사용자가 원하는 정보를 얻거나 작업을 처리하기 위해서는 이러한 웹 페이지들 사이의 이동을 통해 가능하다. 이것이 웹 어플리케이션이 다른 어플리케이션들과 확연히 구분되는 특성인 항해(navigation)를 한다는 점이다. 때문에 항해(navigation) 관계를 결정하는 것은 무엇보다도 중요하다. 이 단계에서 발생되는 산출물은 Navigation Design Diagram 이다.

3.2.4 Page 상세 설계

Page 상세 설계 단계에서는 웹 어플리케이션을 구성하는 각 웹 페이지들의 구체적인 화면 및 기능을 설계한다. 웹 페이지는 그 기능에 따라 정적 페이지와 동적 페이지로 나누어 볼 수 있다. 정적 페이지는 특별한 기능이 없는 디자인 중심의 페이지를 말하며 화면 설계 중심으로 이루어지며, 동적 페이지는 기능 중심의 페이지를 말하며 데이터의 처리, 데이터베이스 연동 등과 같은 기능 설계 중심으로 이루어진다. 이 단계에서 발생되는 산출물은 여러 개의 Page Detail Design Diagram 이다.

3.2.5 코드 생성 및 정제

코드 생성 및 정제 단계에서는 지금까지 단계에서의 산출물들을 이용하여 웹 어플리케이션을 구성하는 웹 페이지를 생성하고, 생성된 웹 페이지의 소스코드를 정제한다. 본 논문에서는 웹 어플리케이션 개발 환경에서의 다이어그램 모델링 결과를 이용하여 소스코드를 자동으로 생성한다. 생성된 웹 페이지 소스코드는 다이어그램을 구성하는 컴포넌트 및 커넥터의 속성과 그들 사이의 관계 등을 통해 자동으로 작성된다. 하지만 이렇게 자동 생성된 소스코드는 완벽하지는 않으므로 시스템 개발자에 의한 소스코드 정제 과정을 거쳐야 한다. 이 단계에서 발생되는 산출물은 웹 어플리케이션을 구성하는 웹 페이지들이다.

3.2.6 운영 및 유지보수

운영 및 유지보수 단계에서는 구축한 웹 어플리케이션을 실제로 운영하고 그에 따른 결함이나 오류를 수정한다. 또한 구축한 웹 어플리케이션에 대한 사용자의 새로운 요구사항이 발생하게 되면, 그에 따라 어플리케이션의 기능을 추가, 수정, 삭제한다.

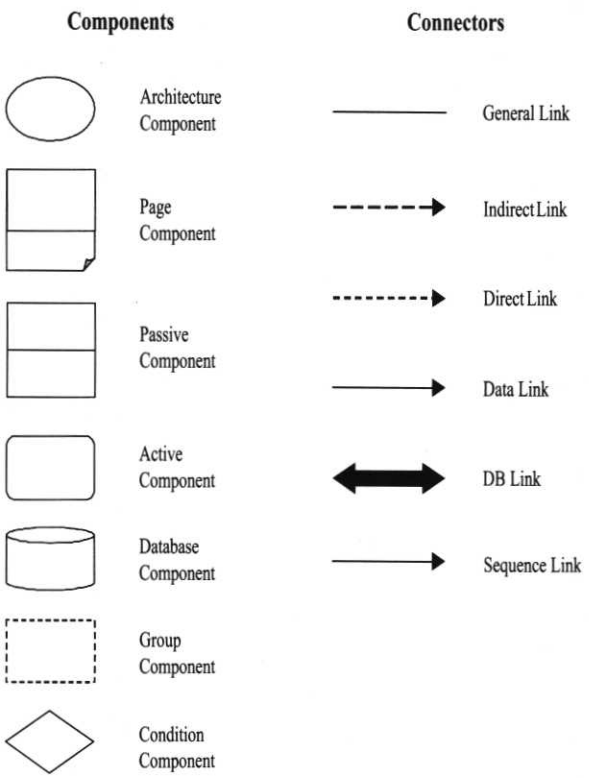
3.3 표기법

웹 어플리케이션을 모델링 하기 위해서는 그 시스템을 구성하는 컴포넌트와 그들 사이의 상호작용을 나타내는 커넥터들이 필요하다. (그림 4)는 본 논문에서 정의한 웹 어플리케이션 모델링을 위한 표기법들을 보여주며, 이들 표기법은 다이어그램에 따라 제한적으로 사용된다.

Architecture Component(⬭)는 Architecture Design Diagram에서 사용되는 컴포넌트로 웹 어플리케이션의 기능을 구조화하여 표현하기 위해 사용된다. 즉, 이는 묶여진 하나의 기능을 나타낸다. Page Component(▭)는 Navigation Design Diagram에서 사용되는 컴포넌트로 웹 어플리케이션을 구성하는 웹 페이지를 표현한다. Passive Component(▭)와 Active Component(▭)는 웹 페이지를 구성하는 정적 및 동적인 컴포넌트를 나타내고, Database Component(⛁)는 데이터 저장소를 나타낸다. Group Component(⬚)는 여러 컴포넌트들을 하나의 특성 또는

기능으로 묶어서 표현하며, Condition Component(◇)는 조건을 명시하기 위해 사용되는 컴포넌트이다. 또한 이들은 모두 Page Detail Design Diagram에서 사용되는 컴포넌트들이다.

General Link(———)는 특별한 의미를 지니지 않고 단순히 두 컴포넌트 사이의 관계 유무만을 나타내며 Architecture Design Diagram에서 사용된다. Indirect Link(- - -▶)와 Data Link(——▶)는 사용자의 클릭에 의한 링크를 말하며, 아무런 데이터의 이동이 없으면 Indirect Link로 나타내고 데이터의 이동이 있으면 Data Link로 표현한다. Direct Link(·····▶)는 프로그램 상에서의 자동 페이지 링크를 나타낸다. 또한 이들은 Navigation Design Diagram에서 사용된다. DB Link(◀▬▶)는 데이터베이스와의 데이터 전송을 나타내고 Sequence Link(——▶)는 두 컴포넌트 사이의 순서 관계를 표현하며, 이들은 Page Detail Design Diagram에서 사용된다.



(그림 4) 웹 어플리케이션 모델링 표기법

3.4 다이어그램

본 논문에서 제시하는 웹 어플리케이션 모델링 환경에는 다음과 같은 다이어그램들이 사용된다.

- Architecture Design Diagram
- Navigation Design Diagram
- Page Detail Design Diagram

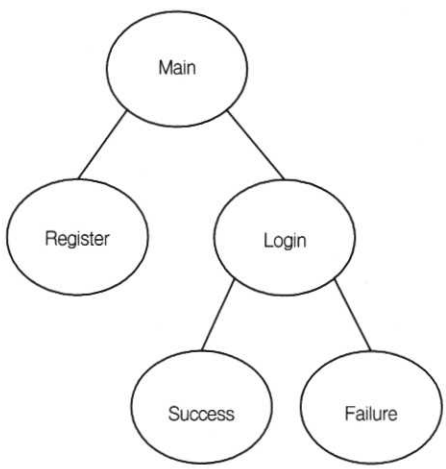
각각의 다이어그램은 웹 어플리케이션을 구조적이면서 동적 및 정적인 기능으로 표현한다. 또한 모든 다이어그램들은 그 역할이 엄격히 분리되어 있지만 서로 연관성을 지니고 있다.

3.4.1 Architecture Design Diagram

Architecture Design Diagram은 기능을 구조화하기 위해 사용되는 다이어그램으로, 웹 어플리케이션이 가지는 기능들 사이의 상하·수평관계를 트리 구조로 표현한다. 여기서는 웹 어플리케이션의 상세한 기능을 표현할 필요는 없으며, 단지 대략적인 기능을 구조화하여 나타낸다. 그리고 이 다이어그램은 다시 Navigation Design Diagram, Page Detail Design Diagram을 통해 구체화된다.

(그림 5)는 로그인 과정을 Architecture Design Diagram으로 표현한 한 예를 보여준 것이다. 이는 사용 가능한 컴포넌트와 커넥터를 이용하여 그들 사이의 관계를 정의하고 각각의 속성을 지정하여 나타낸다. Architecture Design Diagram에서 사용 가능한 컴포넌트 및 커넥터는 다음과 같다.

- Architecture Component
- General Link



(그림 5) Architecture Design Diagram

3.4.2 Navigation Design Diagram

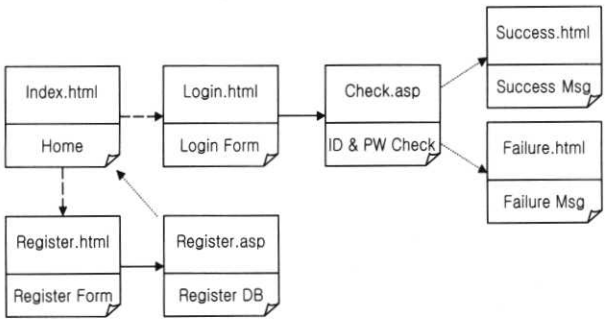
웹 어플리케이션이 다른 어플리케이션과 구분되는 가장 큰 특징은 항해(navigation)를 한다는 것이다. 웹 어플리케이션은 웹 페이지로 구성되어 있으므로, 웹에서 사용자가 정보를 얻거나 작업을 처리하기 위해서는 웹 페이지들 사이의 이동을 통해서 가능하다.

Navigation Design Diagram에서는 웹 어플리케이션을 구성하는 웹 페이지들 사이의 항해 관계를 표현한다. 즉, 웹 페이지들 사이의 링크 관계를 결정한다. 하지만 여기서도 웹 페이지가 가지는 기능을 구체적으로 표현하지는 않으며, 단지 페이지들 사이의 연결 관계 및 데이터 이동을 나타낸

다. 그리고 이 다이어그램에서 표현된 하나의 컴포넌트는 Page Detail Design Diagram에서 구체화된다.

(그림 6)은 (그림 5)의 구조를 Navigation Design Diagram을 이용하여 웹 페이지들 사이의 항해 관계를 표현한 예이다. 이는 사용 가능한 컴포넌트와 커넥터를 이용하여 그들 사이의 관계를 정의하고 그들 각각의 속성을 지정하여 나타낸다. Navigation Design Diagram에서 사용 가능한 컴포넌트 및 커넥터는 다음과 같다.

- Page Component
- Indirect Link
- Direct Link
- Data Link



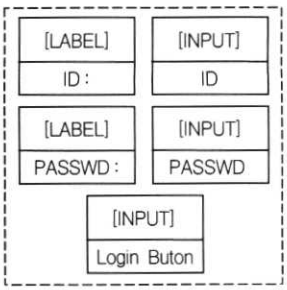
(그림 6) Navigation Design Diagram

3.4.3 Page Detail Design Diagram

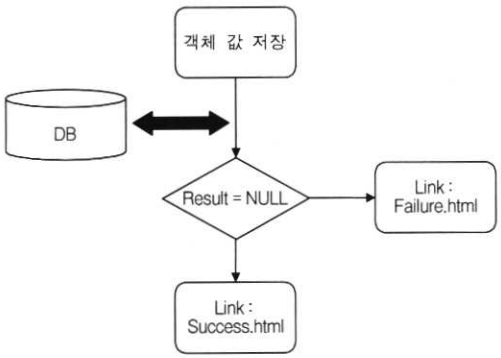
Page Detail Design Diagram은 웹 어플리케이션을 구성하는 각 페이지를 상세하게 설계할 수 있는 다이어그램이다. 여기서 웹 페이지는 정적 페이지와 동적 페이지 그리고 이 두 가지 경우가 복합된 페이지로 나누어 볼 수 있으며, 정적 페이지는 디자인 중심 페이지를 말하고 동적 페이지는 기능 중심 페이지를 말한다.

(그림 7)은 (그림 6)의 컴포넌트를 Page Detail Design Diagram을 이용하여 구체적으로 표현한 예를 보여주며, (그림 7)(a)는 디자인 중심 Page Detail Design Diagram이고 (그림 7)(b)는 기능 중심 Page Detail Design Diagram이다. 이는 사용 가능한 컴포넌트와 커넥터를 이용하여 그들 사이의 관계를 정의하고 각각의 속성을 지정하여 나타낸다. Page Detail Design Diagram에서 사용 가능한 컴포넌트 및 커넥터는 다음과 같다.

- Passive Component
- Active Component
- Database Component
- Group Component
- Condition Component
- DB Link
- Sequence Link



(a) 디자인 중심

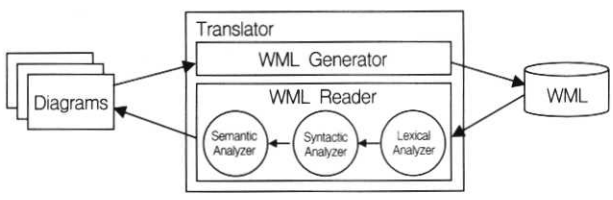


(b) 기능 중심

(그림 7) Page Detail Design Diagram

3.5 모델링 언어

본 논문에서는 WML(Web-application Modeling Language)이라는 웹 어플리케이션 모델링 언어를 정의하였다. WML은 시스템의 기능 및 특성을 정형화하고 구성 및 속성을 구체화한 웹 어플리케이션 모델링 결과의 저장소이다. 즉, 다이어그램을 통한 모델링 결과를 모델링 언어로 나타낸 것이다. 이러한 저장소의 개발은 기존 모델의 재사용을 가져올 수 있다. 이는 다이어그램으로 표현된 웹 어플리케이션 모델링 결과를 바탕으로 자동으로 생성되기도 하고, 시스템 설계자에 의해 직접 작성되기도 한다. 이러한 작업은 본 논문에서 개발된 Translator에 의해 수행되고, 그 구조는 (그림 8)과 같다.



(그림 8)Translator구조

Translator는 두 개의 기능을 가진다. 하나는 다이어그램을 WML로 변환하기 위한 WML Generator이고, 다른 하나는 WML을 다이어그램으로 변환하기 위한 WML Reader이다. (그림 9)는 앞에서 본 (그림 5)의 Architecture Design Diagram을 WML로 변환한 것이다.

```
<Diagram Type = "ArchitectureDesignDiagram" Name = "ADD_Login">
        Comment = "로그인 구조"
        <Component Type = "ArchitectureComponent"
                   Order = "1" X = "1275" Y = "735">
                     Name = "ArchComp 1"
                     Caption = "Main"
                     Comment = "시작 페이지"
                     Parent = " "
                     Child = "ArchComp 2, ArchComp 3"
        </Component>
        <Component Type = "ArchitectureComponent"
                   Order = "2" X = "60" Y = "2205">
                     Name = "ArchComp 2
                     Caption = "Register
                     Comment = "가입"
                     Parent = "ArchComp 1"
                     Child = ""
        </Component>
        <Component Type = "ArchitectureComponent"
                   Order = "3" X = "2550" Y = "2145">
                     Name = "ArchComp 3"
                     Caption = "Login"
                     Comment = "로그인 화면"
                     Parent = "ArchComp 1"
                     Child = "ArchComp 4, ArchComp 5"
        </Component>
        <Component Type = "ArchitectureComponent"
                   Order = "4" X = "1545" Y = "3600">
                     Name = "ArchComp 4"
                     Caption = "Success"
                     Comment = "로그인 성공"
                     Parent = "ArchComp 3"
                     Child = ""
        </Component>
        <Component Type = "ArchitectureComponent"
                   Order = "5" X = "3615" Y = "3630">
                     Name = "ArchComp 5"
                     Caption = "Failure"
                     Comment = "로그인 실패"
                     Parent = "ArchComp 3"
                     Child = ""
        </Component>
        <Connector Type = "GeneralLink" Order = "1" X1 = "1272"
                   Y1 = "1098" X2 = "722" Y2 = "2202">
                     Name = "GnrlLink 1"
                     Relation = "ArchComp 1 → ArchComp 2"
        </Connector>
        <Connector Type = "GeneralLink" Order = "2" X1 = "2622"
                   Y1 = "1098" X2 = "3212" Y2 = "2142">
                     Name = "GnrlLink 2"
Relation = "ArchComp 1 → ArchComp 3"
        </Connector>
        <Connector Type = "GeneralLink" Order = "3" X1 = "2548"
                   Y1 = "2508" X2 = "2208" Y2 = "3598">
                     Name = "GnrlLink 3"
Relation = "ArchComp 3 → ArchComp 4"
        </Connector>
        <Connector Type = "GeneralLink" Order = "4" X1 = "3898"
                   Y1 = "2508" X2 = "4278" Y2 = "3628">
                     Name = "GnrlLink 4"
                     Relation = "ArchComp 3 → ArchComp 5"
        </Connector>
</Diagram>
```

(그림 9) 모델링 언어

3.6 웹 페이지 생성

웹 어플리케이션은 여러 개의 웹 페이지로 구성되며, 본 논문에서 제시한 웹 어플리케이션 개발 환경에서는 설계 단계에서의 산출물인 다이어그램들을 이용하여 자동으로 이러한 웹 페이지를 생성한다. 이는 각 다이어그램을 구성하는 컴포넌트 및 커넥터의 속성과 그들 사이의 관계를 읽어 들이고 이들 사이의 순서관계를 파악하여 소스코드를 생성하게 된다. 이러한 자동 소스코드 생성은 패턴 매칭기법으로 이루어진다. 하지만 이렇게 자동 생성된 소스코드는 완벽하지는 않으므로 시스템 개발자에 의한 소스코드 정제 과정을 거쳐야 한다.

다음 (그림 10)(a)와 (그림 10)(b)는 (그림 7)(a)와 (그림 7)(b)에서 모델링한 결과를 바탕으로 자동으로 생성된 웹 페이지 소스코드를 보여준다.

```
<html>
<head>
</head>
<body>
<form action = "Check.asp" method = "post" name = "frmLogin">
ID : <input type = "text" name = "txtID" value = " " size = "10"
     maxlength = "10"><br>
PASSWD : <input type = "text" name = "txtPasswd"
          value = "" size = "10" maxlength = "10"><br>
<input type = "submit" name = "subLogin" value = "Login"
          size = "10" maxlength = "10"><br>
</form>
</body>
</html>
```

(a) Login.html

```
<%
Id = Request.Form("txtID")
Passwd = Request.Form("txtPasswd")

Set Dbcon = Server.CreateObject("ADODB.Connection")
Dbcon.Open ("DSN = entrance ; UID = sa ; PWD = ")
SQLString = "Select Id From entrance where Id = " & Id
          & " and Passwd = " & Passwd & " "
Set Result = Dbcon.Execute(SQLString)

if Result.EOF then
Response.Redirect("Failure.html")
Dbcon.Close
Set Dbcon = Nothing
else
Response.Redirect("Success.html")
Dbcon.Close
Set Dbcon = Nothing
end if
%>
```

(b) Check.asp

(그림 10) 소스코드 생성

4. 웹 어플리케이션 개발 환경 구현

4.1 구현 환경

본 논문에서는 웹 어플리케이션을 효율적으로 개발하기 위한 웹 어플리케이션 개발 환경을 구현하였다. 다음 <표 1>는 시스템의 구현 환경을 나타낸다.

〈표 1〉 시스템 구현 환경

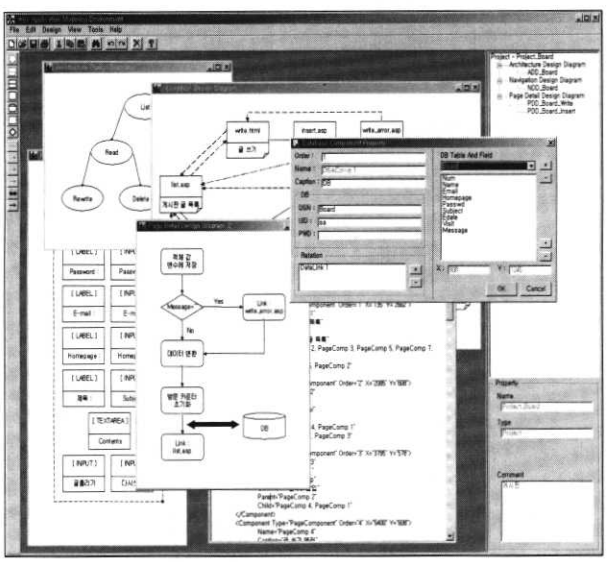
H/W		S/W	
CPU	Pentium Ⅲ 500	OS	Windows 2000 Professional
RAM	256M	개발 툴	Microsoft Visual Basic 6.0
HDD	40G	생성코드	HTML, ASP

4.2 구현 결과

본 논문에서 개발한 웹 어플리케이션 개발 환경은 크게 3가지로 나누어 볼 수 있으며, 그 구성은 Modeling, Translator, Code Generator 이다. 이들의 구현 결과를 보여주기 위해 본 논문에서는 게시판을 예로 적용시켜 보았다. 게시판을 예제로 이용한 이유는 웹 어플리케이션을 개발함에 있어서 사용되는 대부분의 기능을 게시판이 가지고 있으며, 일반적으로 웹 어플리케이션 대부분의 기능이 여기에서 확장된다고 볼 수 있다. 때문에 게시판을 예제로 이용함으로써 웹 어플리케이션의 전반적인 기능을 모두 모델링 할 수 있음을 보여줄 수 있다.

4.2.1 전체 구성

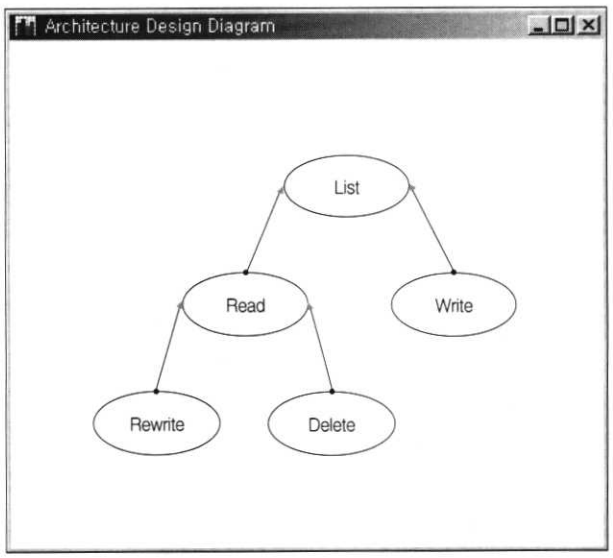
(그림 11)은 본 논문에서 구현한 웹 어플리케이션 개발 환경의 전체화면을 보여준다. 다이어그램을 이용한 모델링과 컴포넌트의 속성 지정, 그리고 모델링 언어로의 변환 및 소스코드 생성에 대한 전체적인 모습을 보인다.



(그림 11) 웹 어플리케이션 개발 환경 전체화면

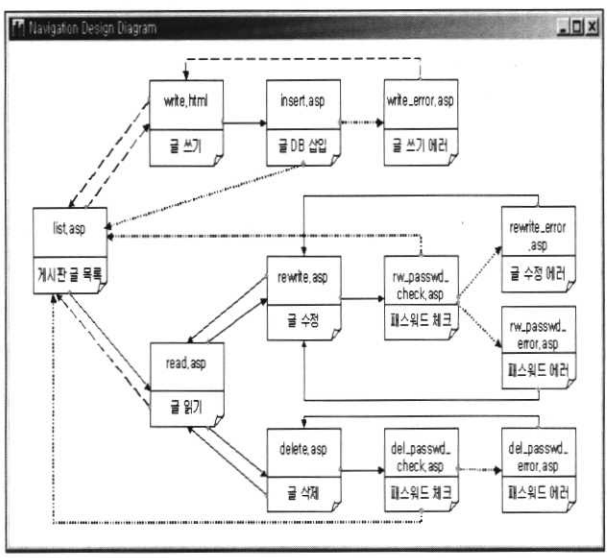
4.2.2 모델링 결과

본 논문에서 예제로 사용한 게시판은 크게 글쓰기, 글읽기, 수정, 삭제 등의 기능이 있으며, 다음과 같은 각 다이어그램별 모델링 결과를 가지게 된다. (그림 12)는 게시판의 전체적인 구조를 표현한 Architecture Design Diagram의 결과화면이다.



(그림 12) Architecture Design Diagram 화면

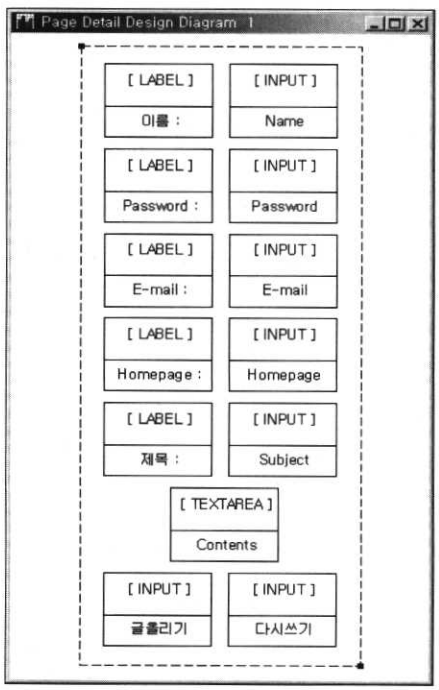
(그림 13)은 (그림 12)에서의 게시판 구조를 항해 관계로 표현한 Navigation Design Diagram의 결과화면을 나타낸다.



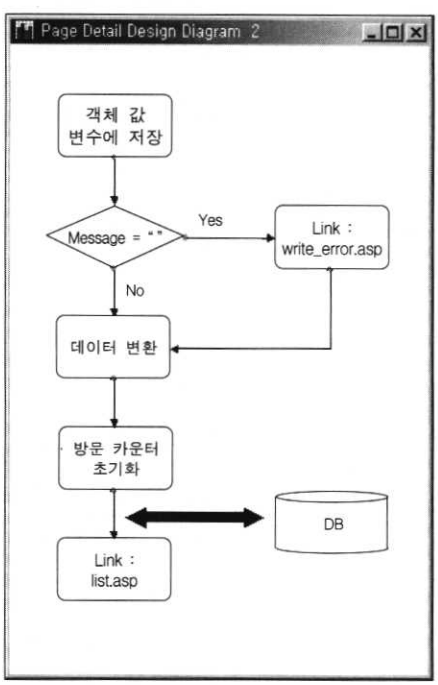
(그림 13) Navigation Design Diagram 화면

(그림 14)와 (그림 15)는 (그림 13)에서의 글쓰기와 글 DB 삽입 컴포넌트를 Page Detail Design Diagram을 이용하여 구체화한 각각의 화면이다. (그림 14)은 write.html 페이지를 나타내며 이는 정적인 디자인 중심 Page Detail

Design Diagram을 보여주며, (그림 15)는 insert.asp 페이지를 나타내며 동적인 기능 중심 Page Detail Design Diagram을 보여준다.



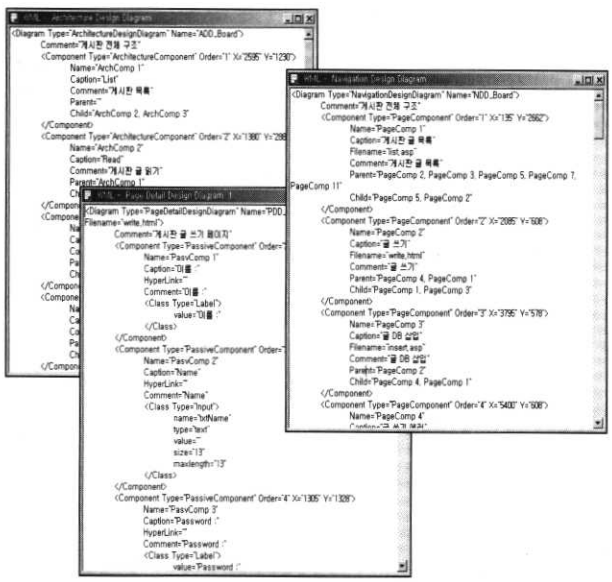
(그림 14) 디자인 중심 Page Detail Design Diagram 화면



(그림 15) 기능 중심 Page Detail Design Diagram 화면

4.2.3 모델링 언어 변환 결과

(그림 16)는 앞에서 살펴본 게시판의 다이어그램별 모델링 결과를 바탕으로 모델링 언어로 변환한 모습을 보여준다.



(그림 16) 모델링 언어 변환 화면

4.2.4 소스코드 생성 결과

(그림 17)은 앞에서 살펴본 게시판의 다이어그램별 모델링 결과를 바탕으로 소스코드를 생성한 결과를 보여준다. 본 논문에서 구현한 시스템은 HTML과 ASP를 기준으로 소스코드를 생성한다.

(그림 17) 소스코드 생성 화면

5. 평가 및 고찰

소프트웨어를 개발함에 있어 분석 및 설계는 그 소프트웨어의 품질을 결정짓는 중요한 작업이다. 따라서 지금까지 이를 위한 많은 개발 기법들이 제시되어 왔으며, 최근

에는 확장된 UML을 통한 설계 기법이 가장 많이 사용되고 있는 실정이다. 하지만 웹 어플리케이션은 일반적으로 일정한 구조를 가지며, 사용되는 기능 역시 매우 제한적이다. 또한 웹 페이지들 사이의 항해를 한다는 점에서 일반 어플리케이션과 크게 구분된다. 이러한 점에서 UML의 확장을 통한 웹 어플리케이션의 모델링은 사용의 어려움과 모델링 결과의 복잡성을 가져온다. 따라서 웹에 적합한 모델링 환경의 필요성이 대두되었고, 이를 위해 본 논문에서는 웹 어플리케이션에 특성화된 모델링 환경을 제시하였다. 뿐만 아니라, UML에서의 소스코드 생성은 주석 및 클래스 정의 정도에서 그쳤으나, WML은 웹에 특성화함으로써 더욱 구체적이고 실질적인 소스코드의 생성을 가능하게 했다. 다음 <표 2>는 특성에 따른 WML과 UML의 상대적 비교 결과를 보여준다.

〈표 2〉 WML과 UML의 비교

특 성	WML	UML	설 명
정확성	높음	높음	WML은 웹 어플리케이션에 특성화되어 개발되었으므로 웹 어플리케이션을 설계함에 있어서의 정확성과 용이성이 높으며, UML은 상대적으로 그 특성이 낮음
용이성	높음	낮음	
재사용성	높음	높음	둘 다 모델링 결과를 위한 저장소의 개발로 기존 모델의 재사용성이 높음
생산성	높음	낮음	WML은 모델링 결과를 바탕으로 실제 사용 가능한 소스코드를 자동으로 생성함으로 개발에서의 생산성이 높음

본 논문에서는 웹 어플리케이션을 효과적으로 개발하기 위해 모델링 과정과 소스코드 생성 과정을 통합한 환경을 제시하였다. 이를 위해 우선 웹 어플리케이션 개발을 위한 프로세스를 정의하였고, 이에 따라 웹 어플리케이션을 설계하고 구현하기 위한 통합된 개발 환경을 구현하였다. 본 논문에서 제시된 웹 어플리케이션 개발 환경의 사용에서 오는 장점은 다음과 같다. 첫째, 웹에 특성화된 모델링 환경을 제공해 줌으로써 어플리케이션 설계 작업을 쉽고 정확하게 해 주며, 설계 결과를 단순화한다. 이로 인해 설계 단계에서 발생되는 오류를 줄일 수 있고, 이것은 결국 소프트웨어의 신뢰성을 높여주는 효과를 가져온다. 둘째, 다이어그램을 통한 웹 어플리케이션 모델링 결과를 WML이라는 모델링 언어로 저장해 둘 수 있으므로 기존 모델의 재사용을 가능하게 한다. 셋째, 모델링 결과를 바탕으로 자동으로 소스코드를 생성해 줌으로써 구현에 드는 시간을 줄일 수 있으므로, 결국 어플리케이션 개발에서의 생산성을 높여준다.

6. 결 론

최근 네트워크의 발전 및 인터넷의 급격한 성장과 함께 기존의 많은 소프트웨어들이 웹 기반으로 그 모습을 변화하고 있다. 이로 인해 웹 어플리케이션의 규모와 복잡성 또한 크게 증가하고 있으며, 웹 어플리케이션 개발에 많은 인력과 비용이 소요되고 있다. 따라서 웹 어플리케이션 개발을 위해 특성화된 환경의 제시가 필요하게 되었다.

본 논문에서는 이러한 필요에 의해 웹 어플리케이션의 개발을 효율적으로 할 수 있는 웹에 특성화된 개발 환경을 제시하였다. 이를 위해 웹 어플리케이션 개발 프로세스를 정의하였고, 이에 따라 설계 및 구현을 할 수 있는 통합된 환경을 개발하였다. 제시된 환경은 Modeling, Translator, Code Generator의 세 가지 모듈로 구성된다. Modeling 단계에서는 Architecture Design Diagram, Navigation Design Diagram, Page Detail Design Diagram을 이용하여 개발하고자 하는 어플리케이션을 설계한다. Tranlstor 단계에서는 다이어그램을 통한 모델링 결과를 모델링 언어로 변환하고, Code Generator 단계에서는 모델링 단계에서의 컴포넌트 속성 및 그들 사이의 관계를 바탕으로 패턴 매칭 기법을 이용하여 실제 소스코드를 생성한다.

결국 본 논문에서 제시한 웹 어플리케이션 개발 환경의 사용은 설계와 구현을 동시에 수행할 수 있어 웹 어플리케이션을 개발함에 있어서의 효율성을 가져오고, 웹에 특성화된 다이어그램과 컴포넌트의 사용으로 설계의 정확성과 단순화를 가져온다. 또한 모델링 결과를 저장할 수 있는 WML이라는 저장소의 개발로 기존 모델의 재사용을 가능하게 하고, 자동 소스코드 생성을 통해 개발 시간을 단축할 수 있어 웹 어플리케이션의 생산성을 향상시킨다.

향후 패턴 매칭 기법을 이용한 자동 소스코드 생성의 방법을 보완하여 보다 정확한 소스코드를 생성하기 위한 방법의 연구가 필요하다.

참 고 문 헌

[1] Pressman, R. S., "Software engineering : a practitioner's approach," McGraw-Hill, 2001.

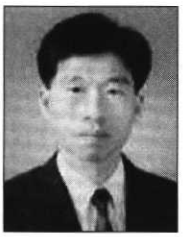
[2] Lin, J., Tsao, H., & Chu, Y., "Object-Oriented Analysis and Design of Web-Based Information Systems," Proceedings of the 8th Annual IEEE International Conference and Workshop on the ECBS, 2001.

[3] Ginige, A., & Murugesan, S., "Web engineering : an introduction," IEEE Multimedia, 8(1), pp.14-18, 2001.

[4] Powell, T. A., Jones, D. L. & Cutts, D. C., "Web site engineering : beyond web page design," Prentice Hall, 1998.

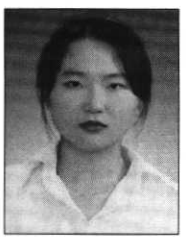
[5] 정병권, 김동수, 송재형, 황종선, "웹 기반 시스템의 분석 및 설계 방법론 개발과 적용", 정보과학회논문지 : 컴퓨팅의 실제, 8(2), pp.155-166, 2002.

[6] Garzotto, F., Paolini, P., & Schwabe, D., "HDM-A Model-Based Approach to Hypertext Application Design," ACM Transactions on Information Systems, 11(1), pp.1-26, 1993.
[7] Isakowitz, T., Stohr, E. A., & Balasubramanian, P., "RM M : A Methodology for Structured Hypermedia Design," CACM, 38(8), pp.34-44, 1995.
[8] Schwabe, D., Rossi, G., & Barbosa, S. D. J., "Systematic Hypermedia Application Design with OOHDM," Proceedings of the ACM International Conference on Hypertext, pp.116-128, 1996.
[9] Eriksson, H. E., Penker, M., "Business Modeling with UML," Wiley, 2000.
[10] Baresi, L., Garzotto, F. & Paolini, P., "Extending UML for Modeling Web Applications," Proceedings of the 34th Hawaii International Conference on System Sciences, Vol.3, 2001.
[11] Schranz, M. W., Weidl, J., "Engineering Complex World Wide Web Services with Jessica and UML," Proceedings of the 33th Hawaii International Conference on System Sciences, Vol.6, 2000.



강 병 도

e-mail : bdkang@daegu.ac.kr
1986년 서울대학교 계산통계학과(이학사)
1988년 서울대학교 대학원 전산과학과(이학석사)
1995년 서울대학교 대학원 전산과학과(이학박사)
1988년~1998년 한국전자통신연구원 선임연구원
1998년~현재 대구대학교 정보통신공학부 교수
관심분야 : 소프트웨어 개발방법론, 소프트웨어 구조, 소프트웨어 프로세스



이 미 경

e-mail : mklee@webmail.daegu.ac.kr
2001년 대구대학교 컴퓨터정보공학부(공학사)
2001년~현재 대구대학교 대학원 컴퓨터정보공학과(석사과정)
관심분야 : 소프트웨어 개발방법론, 소프트웨어 구조, 소프트웨어 프로세스