확장된 UML 모델을 이용한 기민한 웹 애플리케이션 개발 방법론

(An Agile Method for Web Applications Development using Extended UML Model)

이 기 열 † 정 우 성 † 이 춘 우 †
(Keeyoull Lee) (Woosung Jung) (Chunwoo Lee)
이 병 정 †† 김 희 천 ††† 우 치 수 ††††
(Byungjeong Lee) (Heechern Kim) (Chisu Wu)

요 약 웹 애플리케이션은 요구사항이 자주 변경되고, 기존의 소프트웨어와는 다른 아키텍처와 모델을 필요로 하며, 빠른 개발주기 안에 시장에 인도되어야 하는 등 여러 가지 특성으로 인해 기존의 전통적인 소프트웨어 개발 방법론을 적용하기에는 적절치 않다. 본 연구에서는 확장 UML 모델을 이용하여 체계적인 모델링을 수행하면서 동시에 요구사항의 변경에 기민하게 대응할 수 있는 웹 애플리케이션 개발 방법론을 제안한다. 본 방법론에서는 UML을 확장하여 웹 애플리케이션 메타모델을 정의하므로 특정 언어와 기술에 독립적이다. 개발 프로세스는 UML 프로파일 SPEM을 사용하여 기술된다. 프로세스 지원 도구를 통해 프로세스를 실행하고 맞춤화할 수 있다. 웹 애플리케이션을 좀더 체계적이고 효율적으로 모델링할 수 있도록 하해 모델링 도구와 컴포넌트 대화 모델링 도구도 지원한다. 실제 웹 애플리케이션 개발 사례를 통해 프로세스와 웹 애플리케이션 모델의 유용성을 검증한다.

키워드 : 웹 애플리케이션, 기민한 프로세스, UML

Abstract Traditional software development method is not suitable for Web application development because of characteristics of Web application such as frequent requirements change, different architectures and models and quick-to-market delivery. In this paper we propose a Web application development method adaptable to requirements change while we systematically model Web application using extended UML model. The metamodel is independent to specific languages and technologies because we define the metamodel using extended UML model. Proposed process is described by SPEM(Software Process Engineering Metamodel) profile. A process supporting tool execute and customize process. To model Web applications systematically and effectively, a navigation modeling and a component communication modeling tools are provided. In a case study, we show the usefulness of our process and model.

Key words : Web application, Agile Process, UML

· 본 연구는 한국과학재단 특정기초연구(R01-2002-000-00135-0)지원으로 수행되었음

† 학생회원 : 서울대학교 컴퓨터공학부
kylee@selab.snu.ac.kr
wsjung@selab.snu.ac.kr
oniguni@selab.snu.ac.kr
†† 종신회원 : 서울시립대학교 컴퓨터과학부 교수
bjlee@venus.uos.ac.kr
(Corresponding author임)
††† 정 회 원 : 한국방송통신대학교 컴퓨터과학과 교수
hckim@knou.ac.kr
†††† 종신회원 : 서울대학교 컴퓨터공학부 교수
wuchisu@selab.snu.ac.kr

논문접수 : 2006년 8월 18일
심사완료 : 2007년 1월 8일

1. 서 론

인터넷과 월드 와이드 웹이 급속히 대중화되어 웹 기반 시스템과 애플리케이션이 보편화 되었다. 이러한 시스템과 애플리케이션을 만들기 위해서는 반복적이고 체계화된 프로세스와 세부적인 안내 지침, 프로세스에 통합된 지원 도구가 필요하다[1]. 또한 웹 애플리케이션은 요구사항이 자주 변경되고 짧은 개발 주기 내에 시장에 인도되어야 하는 특성을 지닌다. 따라서 체계적인 프로세스와 세부적인 안내지침을 가지는 반면 고객의 요구

사항의 변경에 기민하게 대응할 수 있는 프로세스가 필요하다. 따라서 기존의 모델 기반의 체계적인 개발 방법론[2-9]의 장점을 취하는 반면 인적 요소와 적응성을 중요시하는 기민한 개발 방법론[10-13]의 장점도 가질 수 있는 방법론을 제안한다. 본 방법론에서는 UML 프로파일 SPEM을 사용하여 프로세스를 기술하고, 프로세스를 실행하고 맞춤화할 수 있는 프로세스 지원 도구를 제공한다. 확장 UML 모델에 기반하여 웹 애플리케이션을 좀더 체계적이고 효율적으로 모델링할 수 있는 항해 모델링 도구와 컴포넌트 대화 모델링 도구도 지원한다. 실제 웹 애플리케이션 개발 사례를 통해 프로세스와 웹 애플리케이션 모델의 유용성을 검증한다.

본 논문은 다음과 같이 구성된다. 2장에서는 웹 공학에서 개발 프로세스 및 모델링과 관련된 연구들을 살펴보고, 3장에서는 웹 애플리케이션을 위한 확장 UML 메타모델을 제안한다. 4장에서는 이 모델을 이용하여 실제 웹 애플리케이션을 개발하는 전체 프로세스를 제시하고 5장에서 이를 지원하는 모델링 도구와 프로세스 도구를 보여준다. 6장에서는 사례 연구를 통해 유용성을 검증한다. 마지막으로 7장에서는 결론을 맺는다.

2. 관련연구

2.1 웹 애플리케이션 모델

최근에 모델 기반의 웹 애플리케이션 개발 방법론들이 많이 제안되었다. 특히 전체 개발 프로세스 중에서 설계 방법론에 관한 연구가 활발히 이루어졌다. 기존의 웹 애플리케이션 설계 방법론들은 공통적으로 개념 모델링(Conceptual Modeling), 항해 모델링(Navigation Modeling), 프리젠테이션 모델링(Presentation Modeling)으로 구성된다. 대표적인 연구로 OOHDM[6,7]의 경우에는 개념 모델링, 프리젠테이션 모델링, 추상 인터페이스 설계(Abstract Interface Design), 구현(Implementation) 활동으로 이루어진다. WebML[4,5]에서는 웹 애플리케이션이 요구사항 수집(Requirements Gathering), 데이타 설계(Data Design), 하이퍼텍스트 설계(Hypertext Design), 프리젠테이션 설계(Presentation Design), 사용자/그룹 설계(User/group Design), 맞춤화 설계(Customization Design) 활동을 통해 개발된다. 초기의 설계 방법론들은 OOHDM의 경우 도메인 관련 언어(Domain Specific Language)를 기존의 모델 기반 설계 방법론에 조합하여 웹 애플리케이션을 개발하는 SHDM[14]으로 연구가 확장되고 있다. 또한 최근의 연구로 UWE[15]처럼 확장 UML을 이용하여 웹 애플리케이션을 설계하는 연구도 있다.

웹 애플리케이션이 시장에 빨리 출시되어야 한다는 특성을 고려하면 이러한 설계 방법론들은 너무 많은 산출물과 활동을 가지고 있다. 게다가 기존의 방법론들은 모델링 활동에 집중하여 사용자의 요구사항의 변경에 기민하게 대응하지 못하고, 다음과 같은 한계가 있다[16].

- 비즈니스 프로세스를 모델링하지 못함
- 다양한 추상 수준을 지원하지 못함
- 표준을 사용하지 않음
- 상호작용 측면을 모델링하지 못함
- 동적으로 생성된 콘텐츠를 지원하지 못함

2.2 웹 개발 프로세스

웹 애플리케이션의 특성을 반영하기 위해 기민한 방법론(Agile Method)을 웹 개발에 적용하려는 연구가 있다[11-13]. 이러한 개발 방법론은 전체 개발이 작은 단위의 반복으로 구성되어 매 반복마다 개발 참여자와 고객간의 의사소통이 이루어지며, 이를 통해 요구사항 변경에 기민하게 반응한다. 또한 일반적으로 프로젝트 참여자들의 인적 요소를 강조한다. AWE(Agile Web Engineering)[11-13]은 짧은 전체 개발 주기, 작은 개발팀, 기존 기술의 사용 등에 초점을 맞춘 경량 개발 프로세스이다. 또한 요구상항 분석과 테스팅 및 평가를 권장한다. 그러나 프로세스를 제시할 뿐 지원 도구나 설계 방법론에 대해 구체적인 내용이 없다.

요구사항의 빈번한 변경과 같은 웹 애플리케이션의 특성을 고려하면, XP[17]나 SCRUM[10]과 같은 기민한 프로세스가 웹 공학 도메인에 적당하다. XP는 대화와 피드백의 가치를 중요시하고 SCRUM의 중요한 원칙중 하나가 적응성이다. 이와 같은 기민한 프로세스의 가치와 원칙들은 웹 애플리케이션 개발 프로세스에 중요하지만 기존 웹 공학 개발 방법론에서는 반영되어 있지 않다.

2.3 웹 개발 지원 도구

웹 애플리케이션 지원 도구는 웹 애플리케이션 설계를 지원하는 도구에 관한 연구가 활발히 이루어졌다. RMM에서는 RMCase, OOHDM은 OOHDMWeb 그리고 WebML은 *WebRatio Site development studio 24*라는 도구의 지원을 받는다. 최근의 OOHDM의 확장 방법론인 SHDM의 경우에는 HyperDe라는 도구의 지원을 받는다. 기존의 설계 방법론들은 웹 애플리케이션을 모델링하는 과정을 지원하는 도구에 한정되었지만 최근에는 전체 웹 애플리케이션을 개발하는데 필요한 개발 환경으로 확대되고 있다. 본 연구에서 지원하는 도구도 전체 프로세스를 지원하는 통합 개발 환경을 지향하는 점이 특징이다.

웹 애플리케이션 개발 프로세스를 관리하기 위한 프로세스 지원도구에 관한 연구는 아니지만 일반적인 개발 프로세스를 관리, 실행, 맞춤화할 수 있는 지원도구로는 RPW(Rational Process Workbench)[18]가 있다.

하지만 본 연구에서는 웹 공학 도메인에 특화된 프로세스 지원도구이다.

3. 웹 애플리케이션 메타 모델

본 논문에서 제안하는 프로세스는 페이지 모델링, 항해 모델링, 컴포넌트 대화 모델링 활동을 포함하고 있다. 페이지 모델링은 UML 클래스 다이어그램을 확장한 페이지 다이어그램으로 표현된다. 웹 애플리케이션을 구성하는 기본적인 페이지들 간의 정적인 관계와 페이지를 구성하는 웹 컴포넌트들과 페이지들 간의 정적인 관계를 표현한다. 항해 모델링과 컴포넌트 대화 모델링은 UML의 StateMachine 패키지와 Interaction 패키지를 확장한 항해 모델과 컴포턴트 대화 모델로 표현된다.

3.1 페이지 모델

그림 1은 페이지 모델링을 위한 웹 애플리케이션 메타 모델이다. 웹 애플리케이션을 위한 UML 메타모델에 관한 많은 연구들은 UML 확장 메커니즘을 이용하여 웹 애플리케이션 요소들을 위한 스테레오 타입 등을 추가 하였다. 본 논문에서는 특정 프로그래밍 언어와 기술에 독립적인 웹 애플리케이션 메타모델을 제시한다. 본 모델은 전체 웹 애플리케이션 개발 과정에 통합되어 다른 모델들과 유기적으로 연결된다. 페이지 모델, 항해 모델, 컴포넌트 대화 모델을 모두 합쳐서 하나의 모델을 이루기 때문에 다른 모델들과는 차별성을 가진다.

웹 애플리케이션은 웹 페이지의 집합으로 구성된다. 각각의 웹 페이지는 하나 이상의 웹 컴포넌트를 포함하고 있다. 웹 컴포넌트는 스크립트 컴포넌트, 문서 컴포넌트, 멀티미디어 컴포넌트, 범용 컴포넌트, 폼 컴포넌트로 나뉜다. 스크립트 컴포넌트는 다시 클라이언트 스크립트 컴포넌트, 서버 스크립트 컴포넌트로 나뉜다. 문서 컴포넌트는 다시 가상 문서와 실 문서로 나뉜다. 가상 문서는 서버 스크립트에 의해 생성되는 웹 페이지 문서를 말한다. 클라이언트 스크립트는 자바 스크립트나 비주얼 베이직 스크립트등과 같이 클라이언트 사이드에서 동작하는 스크립트를 말한다. 멀티미디어 컴포넌트는 사진과 같은 이미지, 영화 파일과 같은 동영상, 음악파일과 같은 사운드로 구성된다. 범용 컴포넌트는 자바 애플릿이나 플래시와 같이 독립적으로 동작하는 컴포넌트들을 말한다. 폼 컴포넌트는 사용자로부터 입력을 받은 입력상자, 버튼, 체크박스, 라디오 등을 포함하는 컴포넌트이다.

3.2 항해 모델

항해 모델은 기본적으로 사용자에게 표시되는 페이지를 기준으로 한다. 따라서 항해 모델은 표현 계층을 중심으로 한 전반적인 웹 애플리케이션의 항해를 보여준다.

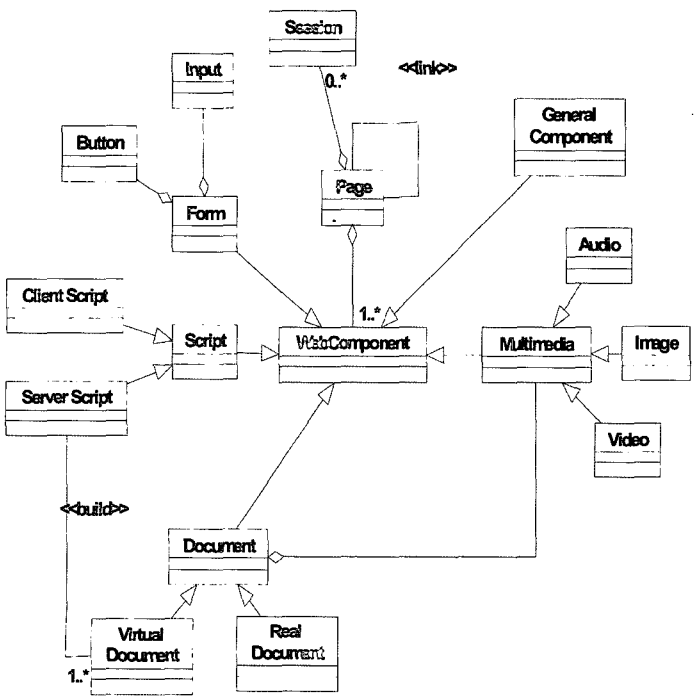

그림 1 웹 애플리케이션 메타모델

그림 2는 UML 상태 기계 다이어그램의 메타모델 중 확장된 부분을 나타낸다. Transition 및 State 클래스는 UML 2.0 상부구조(Superstructure)의 BehaviorStateMachine 패키지에 속한다. Navigation 클래스와 Submit 클래스는 기존의 Transition 클래스를 상속받으며, 각각 일반적인 항해와 데이타 전송이 이루어지는 항해를 의미한다. Submit 클래스의 boundCompID 속성은 이 항해와 연관된 컴포넌트 식별자를 표현하는 것으로 ComponentID 타입을 갖는다. PageView 클래스는 페이지 모델의 실문서에 대응되는 뷰를 나타낸다. VirtualPageView 클래스는 서버 스크립트에 의해 생성되는 가상 문서에 대응되는 페이지의 뷰이다. 이러한 페이지는 물리적으로 존재하지 않으나 사용자에게 표시되므로 항해 모델에 나타낸다. Page의 이름은 실재하는 페이지 파일의 이름을 붙인다. VirtualPageView의 경우 이 페이지를 생성하는 서버 스크립트 페이지의 이름을 변형하여 붙인다.

표 1 항해 모델의 노드 타입과 경로 타입의 표기법

노드 타입	아이콘	경로 타입	아이콘
PageView	Ⓟ	Navigation	
VirtualPageView	Ⓥ	Submit	

표 1은 확장한 UML 상태 기계 다이어그램의 노드(node) 타입과 경로(path) 타입의 표기법을 보여준다. 실제 항해 모델링과 컴포넌트 대화 모델링의 예는 5장 사례연구에서 볼 수 있다.

3.3 컴포넌트 대화 모델

컴포넌트 대화 모델은 웹 애플리케이션의 표현 계층과 비즈니스 논리 계층 간의 연결 지점을 보여주므로 본 연구에서 제안하는 프로세스에서 컴포넌트의 구현을 위한 밑그림이 된다. 컴포넌트 대화 모델은 UML 2.0의 시퀀스 다이어그램을 확장하여 표현한다. 시퀀스 다이어그램은 단일한 유즈 케이스에 종속적이다. 따라서 웹 애플리케이션의 표현 계층 및 비즈니스 논리 계층이 연결

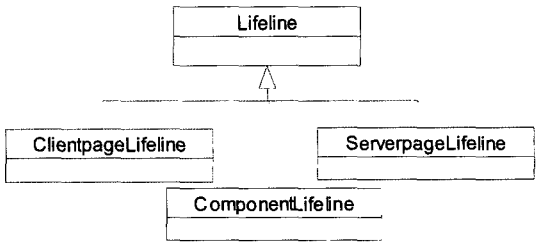

그림 3 컴포넌트 대화 모델을 위한 UML 시퀀스 다이어그램의 확장

표 2 컴포넌트 대화 모델의 노드 타입과 표기법

노드 타입	아이콘
ClientpageLifeline	cpName P_C
ServerpageLifeline	spName P_S
ComponentLifeline	cmpName C

되는 지점별로 모델을 작성한다. 각각의 모델이 만들어진 후 식별된 컴포넌트의 기능에 따라 컴포넌트 그룹화 및 통합이 이루어질 수 있다. 그림 3은 UML 시퀀스 다이어그램 메타 모델의 확장된 부분을 나타낸다. Lifeline 클래스를 클라이언트 페이지, 서버 페이지 및 컴포넌트를 위한 Lifeline 클래스로 확장하였다. Lifeline 클래스는 BasicInterations 패키지에 속한다. 표 2는 확장된 노드의 표기법이다. 이외의 컴포넌트 대화 모델 요소는 시퀀스 다이어그램의 요소 정의를 따른다.

4. 웹 애플리케이션 개발 프로세스

웹 애플리케이션은 인지(Perception) 단계(Phase)와 제작(Production) 단계로 구성된 반복적이고 점증적인 프로세스에 따라 개발된다. 인지 단계는 웹 애플리케이션의 기초 분석 활동을 포함하며, 프로젝트의 초반에 가장 먼저 수행된다. 제작 단계는 개발 주기의 기본 단위로서 웹 애플리케이션의 기능을 구현하는 것을 목표로

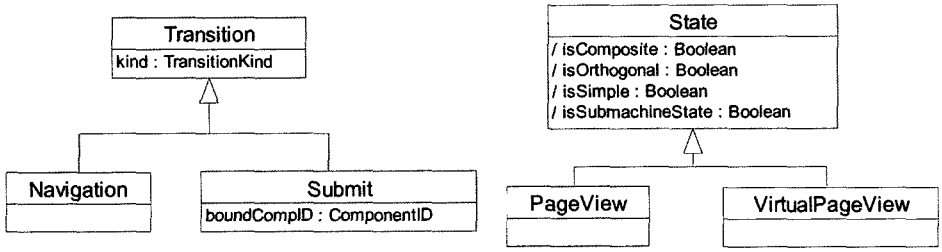

그림 2 항해 모델을 위한 UML 상태 기계 다이어그램의 확장

한다. 개발자는 제작 단계를 반복하면서 웹 애플리케이션을 완성한다. 그림 4는 제안하는 프로세스의 개관을 보여준다. 본 프로세스는 확장성과 호환성을 위하여 OMG의 SPEM 1.1[19]에 기반하여 기술하였다. SPEM은 소프트웨어 개발 프로세스를 표현하기 위한 UML 프로파일이다. SPEM은 프로세스를 모델링하기 위하여 객체를 중심으로 접근하고, MOF 기반의 완전한 메타모델을 제공한다. 이러한 접근 덕분에 UML 도구와 MOF 기반 도구나 저장소 사이의 교환이 가능하다.

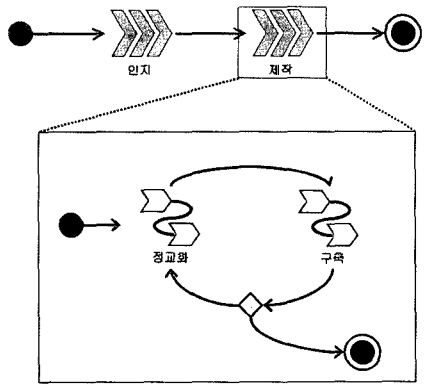

그림 4 전체 프로세스 개관

본 프로세스에서는 웹 애플리케이션 설계 작업시 확장 UML 메타모델에 기반하여 웹 애플리케이션을 설계한다. UML 기본 요소만으로는 웹 애플리케이션을 모델링하기에 충분하지 않기 때문에 UML 확장 메카니즘을 사용하여 웹 애플리케이션의 특성을 반영하도록 확장하였다. 그리고 본 프로세스는 웹 애플리케이션 개발의 잦은 요구사항 변경을 기민하게 반영하기 위해 각 개발 주기의 초반에 고객 확인 절차를 배치하여 초기 설계 사항에 대한 고객의 피드백을 얻는다. 또한 각 개발 주기가 빠르게 진행되므로, 개발 주기 중 변경된 요구사항은 비교적 빨리 반영된다. 그리고 협업(collaboration), 개발자 개개인의 능력(competence of team members) 등 기민한 프로세스(agile process)에 일반적으로 기대되는 인적 요인(human factor)[20]에 대한 고려가 이루어졌다. 결과적으로, 본 프로세스는 확장 UML 메타모델에 기반한 모델링 작업을 포함한 기민한 프로세스를 지향한다.

4.1 역할(Roles)

프로세스 역할은 개발에 참여하는 인력의 특정 역할을 의미한다. 따라서 한 개발자가 다양한 프로세스 역할을 맡거나 한 프로세스 역할에 여러 인력이 투입될 수 있다. 웹 공학 개발 프로세스에서 필요한 역할을 다음과 같이 정의하였다.

- 프로젝트 관리자(Project Manager): 프로세스 역할과 프로젝트 전반에 걸친 활동, 산출물 등을 관리하는 역할
- 소프트웨어 분석가(Software Analyst): 웹 애플리케이션의 기능/비기능적 특성을 분석하여 설계에 반영하는 역할
- UI 개발자(UI Developer): 페이지 저작 및 UI 관련 콘텐츠를 개발하는 역할
- 컴포넌트 개발자(Component Developer): 비즈니스 논리 단의 컴포넌트 제작 및 재사용, 명세 작성 등을 담당하는 역할
- DB 개발자(Database Developer): 웹 애플리케이션을 위한 데이타베이스 시스템을 개발하는 역할
- 아키텍트(Architect): 초기 아키텍처 및 컴포넌트 아키텍처를 설계하는 역할
- 시험자(Tester): 시험 사례 생성 및 시험 활동을 수행하는 역할
- 도메인 전문가(Domain Expert): 웹 애플리케이션의 해당 도메인 기반 지식 전문가로 초기 설계 시에 소프트웨어 분석가를 보조하는 역할
- 고객(Client): 웹 애플리케이션의 제작 의뢰인으로, UI 프로토타입 및 스토리보드에 대한 확인 작업을 수행하는 역할

4.2 웹 애플리케이션 개발 단계

4.2.1 인지 단계(Perception Phase)

프로젝트가 시작되면 먼저 인지 단계가 수행된다. 그림 5는 인지 단계에서 수행되는 활동들을 보여주는 활동 다이어그램(activity diagram)이다. 인지 단계에서는 웹 애플리케이션의 기초 분석 작업이 수행되며 그 결과로 사용자 요구사항, 초기 아키텍처 등이 작성된다.

우선 소프트웨어 분석가는 고객으로부터 문제 기술서(problem statements)를 건네받아 이를 바탕으로 사용자 요구사항을 작성한다. 그리고 도메인 전문가와 협업하여 페이지모델을 작성한다. 작성된 페이지 모델과 사용자 요구사항을 고려하여, 아키텍트는 웹 애플리케이션의 초기 아키텍처를 설계한다. 다음은 본 단계에서 수행되는 활동들에 대한 상세 기술이다.

◦ 사용자 요구사항 작성

소프트웨어 분석가는 고객의 문제 기술서를 바탕으로 기능/비기능적 요구사항을 추출하여 이를 구조적으로 기술한다. 기능적 요구사항은 웹 애플리케이션이 사용자에게 제공하는 기능을 기술하며 프로젝트의 개발주기별 목표 설정을 위한 기초 자료가 된다. 비기능적 요구사항은 보안, 성능, 신뢰성 등 웹 애플리케이션에 기대되는 정성적인 특질을 포함하며 이후 아키텍처 설계를 통해 시스템에 반영된다.

◦ 페이지 모델링

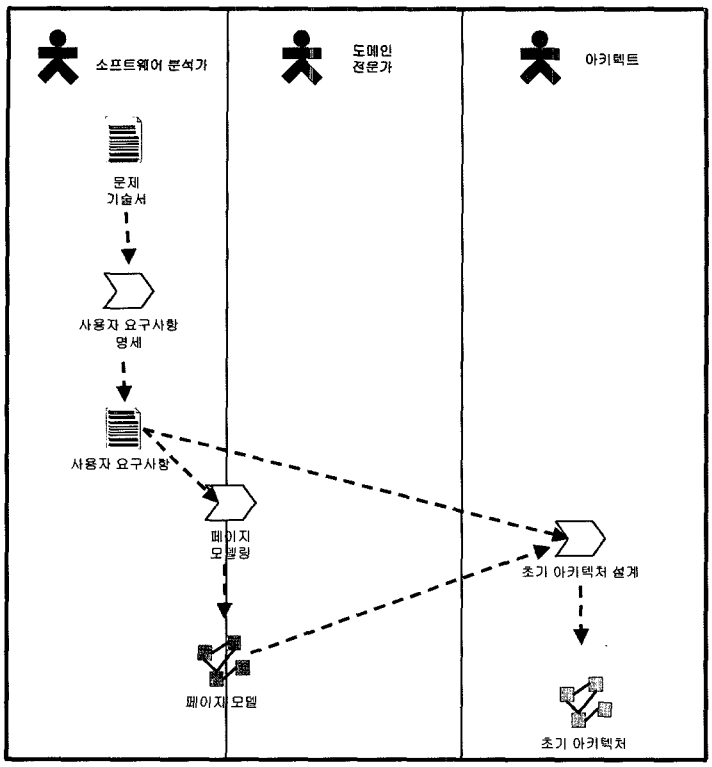

그림 5 인지 단계 활동 다이어그램

작성된 사용자 요구사항을 바탕으로, 소프트웨어 분석가는 페이지 모델을 작성한다. 웹 애플리케이션의 가장 기본이 되는 요소는 페이지이므로 페이지 단위로 생각하는 것이 자연스럽다. 필요에 따라 도메인 전문가는 페이지 모델이 해당 도메인의 비즈니스 모델 정보를 잘 반영할 수 있도록 소프트웨어 분석가에게 조언하는 역할을 담당한다.

• 초기 아키텍처 설계

아키텍트는 사용자 요구사항과 페이지 모델을 고려하여 웹 애플리케이션의 초기 아키텍처를 설계한다. 초기 아키텍처는 구체적인 컴포넌트나 서브시스템에 대한 정보가 없는 상태에서 추상적인 설계 결정을 반영한다. 아키텍처를 수립할 때, 아키텍트는 사용자 요구사항에 기술된 비기능적 요구사항을 반드시 반영해야 한다. 초기 아키텍처는 UML 2.0 컴포넌트 다이어그램으로 표현된다.

4.2.2 제작 단계

제작 단계는 상세 설계 및 구현, 시험 및 통합 활동 등을 포함하며, 개발 주기의 기본 단위로써 초기에 설정된 개발 목표에 따라 반복적으로 수행된다. 제작 단계는 정교화(Sophistication) 부단계(Sub Phase) 및 구축(Building) 부단계로 구성된다.

(1) 정교화 부단계

정교화 부단계(그림 6)는 설계 활동 및 UI 프로토타이핑, DB 시스템 제작 활동 등으로 구성된다. 각각의 활동들은 의존성 분석을 통해 일부 병렬 수행될 수 있도록 배치되었다.

정교화 부단계의 설계 활동의 최소 단위는 컴포넌트로서, 컴포넌트 수준의 고 수준 결정들은 매 개발주기를 거쳐 수정되고 정제된다. 다음은 정교화 부단계에서 수행되는 활동들에 대한 상세 기술이다.

• 목표 설정

소프트웨어 분석가는 사용자 요구사항을 참조하여 해당 개발 주기의 목표를 설정한다. 개발 목표는 기본적으로 인지 단계에서 산출된 사용자 요구사항 중 일부를 선별하여 작성된다. 만약 전 개발주기에서 개발을 목표하였으나 미 구현된 부분이 있거나 변경된 요구사항이 있으면 현 개발주기의 목표에 이들을 포함시킨다. 개발 목표는 한 개발 주기가 일주일 정도의 짧은 시간에 완료될 수 있도록 적은 수의 기능적 요구사항과 그와 관련한 비기능적 요구사항으로 제한한다.

• 수용 시험 사례 작성

시험자는 개발 주기의 목표를 바탕으로, 해당 개발 주기에서 만들어질 웹 애플리케이션 부분에 대한 수용 시험 사례(acceptance test case)를 작성한다. 본 프로세

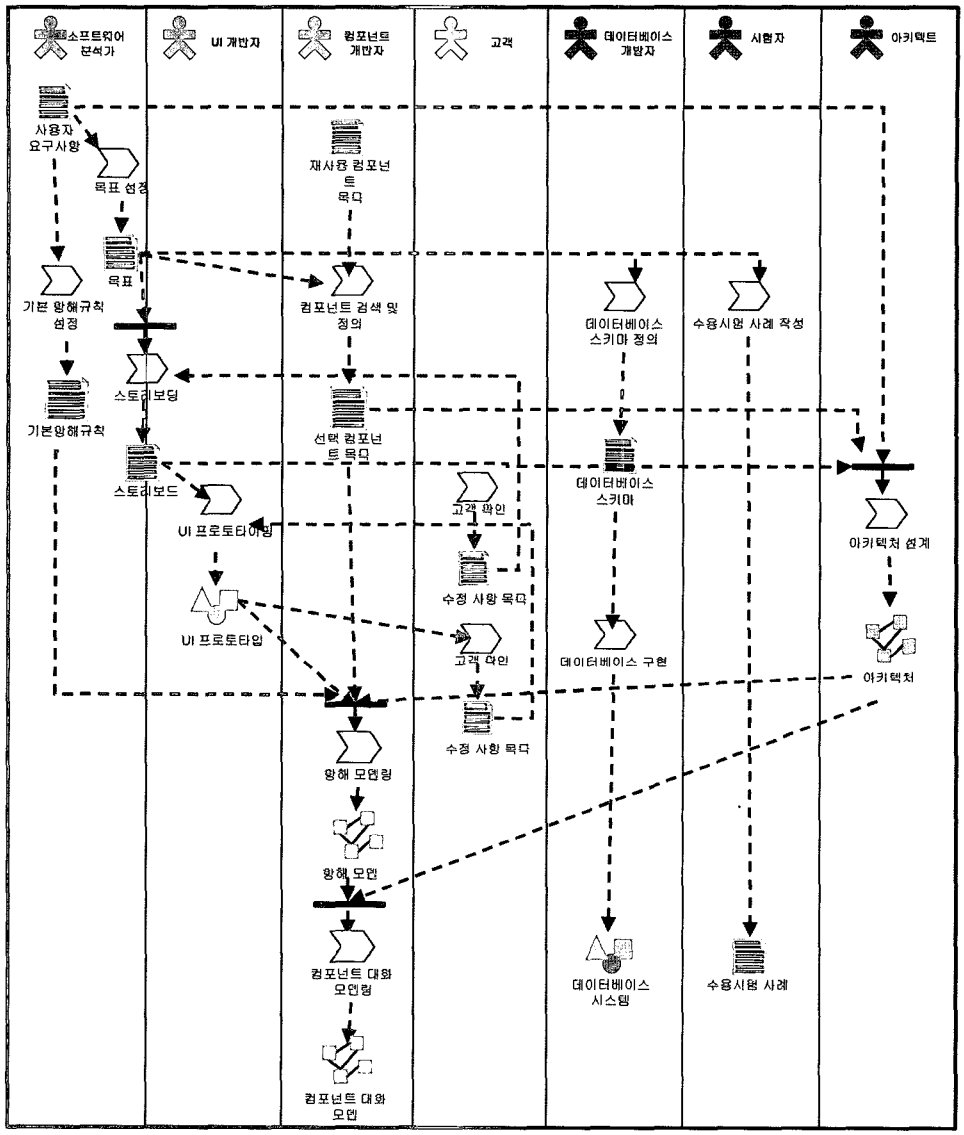

그림 6 정교화 부단계의 활동 다이어그램

스에서는 매 개발주기마다 개발 완료되는 웹 애플리케이션의 부분을 바탕으로 최종 수용 시험이 이루어지며, 이때 본 활동에서 만들어진 시험 사례가 사용된다. 수용 시험 사례는 일반적인 시험 사례의 형식으로 기재된다.

◦ 컴포넌트 검색 및 정의

컴포넌트 개발자는 해당 개발주기의 목표를 검토하여 재사용 컴포넌트 목록에서 필요한 컴포넌트를 검색한다. 재사용 컴포넌트 중 해당 개발주기에 사용될 컴포넌트는 선택 컴포넌트 목록에 기재한다.

재사용 컴포넌트 목록에서 필요한 컴포넌트가 발견되지 않을 경우, 이러한 컴포넌트는 향후 구축 부단계에서 구현되어야 한다. 그리고 해당 개발주기에서 새로 개발된 컴포넌트는 다음 개발주기, 혹은 다른 프로젝트에서 재사용될 수 있다. 따라서 컴포넌트 개발자는 이러한 신규 개발 컴포넌트의 명세를 작성하여 재사용 컴포넌트와 선택 컴포넌트 목록의 양쪽에 추가한다.

• 기본 항해 규칙 설정

소프트웨어 분석가는 사용자 요구사항을 검토하여 기본 항해 규칙을 추출한다. 항해 규칙은 단일한 요구사항에서 추출되므로, 전체 항해를 보여주지 않으며 일부 항해 제약 사항만을 반영한다. 예를 들어 '회원 정보 페이지는 로그인 상태에서만 열람이 가능하다'라는 요구사항

이 있을 경우 회원 정보 페이지로의 항해에는 반드시 회원 로그인 세션 정보가 필요하다는 것을 알 수 있다. 이러한 항해 정보를 정형화하여 기본 항해 규칙을 설정한다.

기본 항해 규칙 간에도 불일치점이 발생할 수 있다. 따라서 정형화한 항해 규칙간의 일관성을 먼저 검사하고, 불일치하는 경우 요구사항을 수정하여 문맥을 통일하여야 한다. 일관성을 갖춘 기본 항해 규칙은 이후 항해 설계 단계에서 항해 모델의 검증 용도로 사용된다. 기본 항해 규칙의 검증을 통하여 사용자 요구사항과 항해 모델간의 추적성(traceability)이 일정 정도 보장된다.

• 스토리보딩 (Storyboarding)

UI 개발자와 소프트웨어 분석가는 해당 개발주기의 개발 웹 애플리케이션의 전반적인 흐름을 보여주는 스토리보드를 작성한다. 스토리보드는 각 페이지의 개략적인 콘텐츠 구조와 사용되는 컴포넌트 식별자, 그리고 페이지 간의 항해를 보여주는 산출물로 웹 애플리케이션의 기능적 요구사항 및 항해구조 표현에 중점을 둔다. 이 중 컴포넌트 식별자는 선택 컴포넌트 목록의 식별자와 동일하다. 본 프로세스에서는 해당 개발주기에서 개발되는 웹 애플리케이션의 모든 페이지를 스토리보드 상의 페이지 형태로 표현한다.

• 고객 확인 (스토리보드)

완성된 스토리보드는 고객에 의한 확인 절차를 거친다. 고객은 스토리보드에서 보여주는 항해 구조 및 웹 애플리케이션의 구동 방식을 검토하고, 이에 대한 의견을 제시한다. 만약 고객의 원래 의도와 다르게 작성된 부분이 있을 경우, 고객은 이에 대한 수정 목록을 작성하여 개발자에게 전달한다. 이러한 수정 목록의 항목 여부에 따라, 소프트웨어 분석가와 UI 개발자는 고객의 의사를 반영하여 스토리보딩을 다시 수행하여 고객에게 확인 작업을 요청한다. 최종적으로 고객의 수정 목록이 더 이상 없을 경우에 한하여 스토리보딩과 고객 확인 활동으로 이루어진 반복(iteration)이 종료된다.

• 항해 모델링

스토리보드로부터 항해 정보를 추출하여 항해 모델을 작성한다. 항해 모델은 본 연구에서 제안하는 '웹 애플리케이션을 위한 UML 모델' 중 항해 다이어그램을 사용하여 작성한다. 항해 모델을 만들면서, 개발 웹 애플리케이션의 항해 구조가 기본 항해 규칙을 위반하지 않는지를 검증한다. 만약 기본 항해 규칙을 위반하는 항해가 발견될 경우, 스토리보드에 대한 수정 작업이 이루어진다.

본 연구에서는 항해 모델링을 지원하는 도구를 제공한다. 이 도구를 사용함으로써 앞서 스토리보드에서 추출해낸 정보를 바탕으로 항해 모델 작성 및 항해 규칙 검증 작업 등을 일부 자동으로 수행할 수 있다.

• UI 프로토타이핑

UI 개발자는 스토리보드를 바탕으로 UI 프로토타입을 작성한다. 기본적으로 UI 프로토타입은 스토리보드를 구성하는 페이지 각각에 해당하는 클라이언트 페이지들로 구성된다. 비즈니스 논리단의 컴포넌트와 상호 작용하는 항해의 경우, 컴포넌트를 스텁(stub)으로 대체하여 개발한다. 스토리보드가 항해 구조 및 컴포넌트 사용 측면에 대한 결정을 보여준다면, UI 프로토타입은 UI 관련 결정들을 보여준다. 즉, 구체적인 UI의 설계와 콘텐츠를 포함한다. 이를 위하여 UI 개발자는 콘텐츠를 직접 개발하거나 아웃소싱(outsourcing)할 수 있다. UI 콘텐츠는 텍스트, 그림, 멀티미디어 파일 및 플래쉬(Macromedia Flash) 등을 포괄한다.

완성된 UI 프로토타입은 버려지지 않고 이후 작업을 통해 실제 웹 애플리케이션의 UI가 된다. 각 페이지와 관련하여 스텁으로 구현된 컴포넌트들은 이후 구축 부단계에서 실제 컴포넌트로 대치된다. 또한 서버 페이지의 경우도 컴포넌트와 마찬가지로 구축 부단계를 통해 구현되어, UI 프로토타입의 해당 클라이언트 페이지를 대체하게 된다.

• 고객 확인 (UI 프로토타입)

완성된 UI 프로토타입에 대한 고객 확인 활동이 이루어진다. 스토리보드의 고객 확인 작업과 마찬가지로 수정 사항을 반영한 UI 프로토타이핑 재작업이 이루어지고, UI 프로토타입의 수정 사항에 대한 고객의 확인 작업이 다시 수행된다.

본 프로세스에서 두 번 수행되는 고객 확인 작업은 전체적인 수행 절차가 유사하나, 그 목적이 상이하다. 스토리보드에 대한 고객 확인 활동은 주로 항해 및 컴포넌트 사용 등에 대한 검토인데 반해 UI 프로토타입에 대한 고객 확인 활동은 UI 결정에 대한 검토이다.

• 아키텍처 설계

아키텍트는 해당 개발주기의 개발 웹 애플리케이션의 컴포넌트 수준 아키텍처를 설계한다. 컴포넌트 수준 아키텍처는 최소 구성단위가 컴포넌트이며, 컴포넌트의 집합으로 부시스템(subsystem)을 정의하여 이들 간의 상호 작용을 설계하는 것이다. 이 과정에서 아키텍트는 사용자 요구사항의 비기능적 요구사항을 종합, 시스템에 요구되는 비기능적 특성을 반영하여 다양한 디자인 패턴을 적용할 수 있다. 아키텍처는 초기 아키텍처와 마찬가지로 UML 2.0 컴포넌트 다이어그램으로 표현된다.

컴포넌트 간 상호 작용에 관련한 정보는 스토리보드, 재사용 컴포넌트 목록 및 선택 컴포넌트 목록 등에서 추출한다. 이들 중 재사용 컴포넌트 목록 및 선택 컴포넌트 목록에 기재된 각 컴포넌트의 인터페이스는 필요

에 따라 수정될 수 있다.

아키텍처는 개발주기간의 개발 일관성을 유지하는 중요한 기능을 수행한다. 즉, 이전 개발주기에서 작성된 아키텍처는 전체 개발주기가 종료되는 시점까지 유지/관리되며, 개발의 진행에 따라 새로운 컴포넌트가 추가되거나 기존 아키텍처가 수정된다. 이러한 컴포넌트 수준의 설계 결정을 유지함으로써 본 프레임워크는 모델 기반 방법론(model-driven methodology)의 장점을 이용한다.

- 컴포넌트 대화 모델링(Component Communication Modeling)

컴포넌트 개발자는 항해 모델의 <<Submit>> 전이(transition) 각각에 대해 컴포넌트 대화 모델을 작성한다. 컴포넌트 대화 모델은 본 연구에서 제안한 '웹 애플리케이션을 위한 UML 모델' 중 컴포넌트 대화 다이어그램을 이용하여 작성한다. 컴포넌트 대화 다이어그램은 UML 시퀀스 다이어그램을 확장한 것으로, <<Submit>> 전이에 관련하는 페이지, 컴포넌트 간의 대화를 표현한다. 따라서 컴포넌트 사용 정보를 명시한 스토리보드와 각각의 항해 종류를 기술한 항해 모델 및 아키텍처를 참조한다. 컴포넌트 대화 모델을 종합하면 신규 개발 컴포넌트의 인터페이스를 정할 수 있다. 이렇게 결정된 컴포넌트 인터페이스는 구축 부단계에서의 컴포넌트 개발을 위한 밑그림이 된다.

- 데이타베이스 스키마 설계/데이타베이스 구현

데이타베이스 개발자는 데이타베이스 시스템의 스키마 설계 및 구현 활동을 수행한다. 데이타베이스 스키마 설계 및 구현 활동은 정교화 부단계에서 완료되어 구축 부단계의 컴포넌트 개발에 직접적으로 사용될 수 있도록 한다.

(2) 구축 부단계

구축 부단계는 실제적인 컴포넌트 구현 및 통합, 시험 활동 등으로 이루어진다. 구축 부단계의 모든 활동은 시험 기반 방법론[21]에 입각하여 진행된다. 컴포넌트 및 통합 웹 애플리케이션은 시험 사례를 먼저 작성한 후에 이를 만족하는 구현을 추가하는 방식으로 만들어진다. 그림 7은 구축 부단계의 전체 활동을 보여준다. 다음은 구축 부단계의 활동들에 대한 상세 기술이다.

- 컴포넌트 구현 및 통합

컴포넌트 개발자는 정교화 부단계에서 산출된 컴포넌트 대화 모델을 기반으로 컴포넌트를 개발한다. 컴포넌트 개발은 기본적으로 시험 기반 원칙에 따라 수행한다. 만들어진 컴포넌트 들은 정교화 부단계에서 작성된 컴포넌트 수준 아키텍처에 따라 통합된다. 컴포넌트 간의 통합 작업에도 시험 기반 방법론을 적용한다. 따라서 각 컴포넌트 간의 결합이 이루어질 때마다 이를 위한 시험

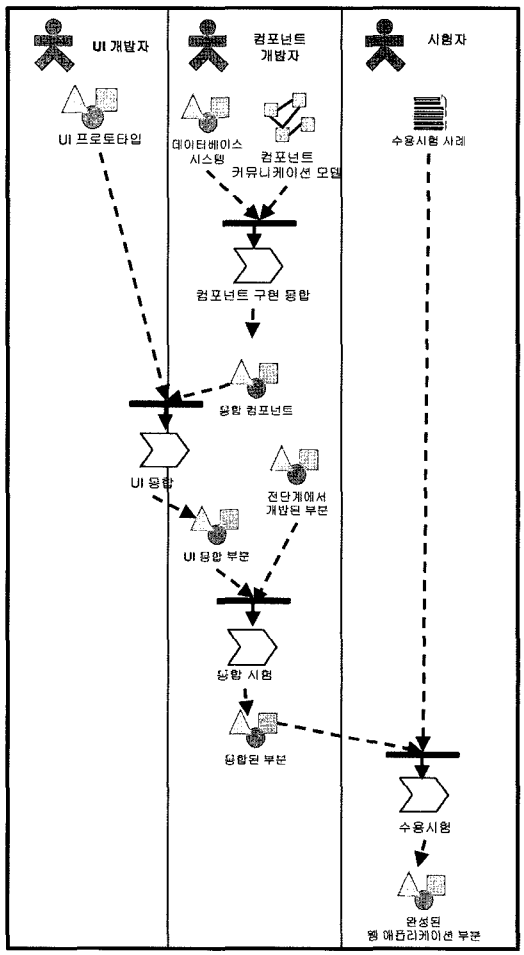

그림 7 구축 부단계 활동 다이어그램

사례가 먼저 작성되어야 하며, 이 시험 사례를 통과한 경우에 한해서 다음 작업이 이루어진다. 재사용 컴포넌트 및 신규 컴포넌트 들이 모두 통합되면 해당 개발주기의 웹 애플리케이션 부분의 비즈니스 논리 층위가 완성된다.

- UI 통합

컴포넌트 구현 및 통합 활동에서 비즈니스 논리 층위가 완성되면, 정교화 부단계에서 만들어진 UI 프로토타입과 비즈니스 논리 컴포넌트 간의 통합 작업이 이루어진다. UI 프로토타입에서 스텁으로 위치하던 컴포넌트는 이미 개발/통합이 완료된 상태이므로, 이들을 실제 컴포넌트로 대치시킨다. 각 컴포넌트 대치 과정은 시험 기반 방법론에 따라, 먼저 시험 사례를 작성하고 이를 위한 통합 작업을 수행하는 순으로 진행한다.

- 통합 시험

해당 개발주기에서 개발된 웹 애플리케이션과 이전

개발주기들을 거쳐 만들어진 웹 애플리케이션을 통합한다. 통합 과정은 구축 부단계의 다른 활동들과 마찬가지로 시험 기반 방법론의 원칙을 적용하여 수행된다. 즉, 통합된 기능(functionality)에 대한 시험 사례를 먼저 만들고 난 후에 통합된 부분이 각 시험 사례를 통과하는 경우에 한정하여 순차적으로 통합 작업을 행한다.

• 수용 시험

시험자는 인지 단계에서 만들어진 수용 시험 사례를 이용하여 통합된 웹 애플리케이션을 시험한다. 시험 결과 통과되지 못한 항목은 다음 개발주기의 목표에 반영한다.

5. 지원 도구

5.1 프로세스 관리 도구

프로세스 관리 도구는 UML 도구를 통해 작성된 프로세스 모델을 XMI로 변환시켜 시스템에 적재하여 저장한다. 이 모델은 이종 플랫폼 간에 공유를 가능하게 한다. 프로젝트 수행 시에는 특정 프로세스를 선택하고 인스턴스를 생성시켜 미리 정의된 가이드라인에 따라 프로세스를 수행할 수 있다. 이러한 과정에서 프로세스 역할 별로 관련 산출물을 생성하고, 필요한 도구를 사용할 수 있다. 본 논문에서 제안한 프로세스도 이 도구를 이용하여 생성하였고 이 도구는 각 프로젝트의 특성에 따라 제안된 프로세스를 최적화할 수 있도록 도와준다.

그림 8은 프로젝트를 수행하면서 따라야 할 프로세스 모델로 새로운 프로세스를 선택하는 새 프로젝트 마법사 화면이다. 선택 후에 프로젝트에 대한 간략한 설명을 추가하면 실행 가능한 프로젝트가 생성되고, 실행시키면 단계 및 활동, 스텝 정보들이 산출물과 연관되어 구조적으로 확인 가능한 상태가 된다.

그림 9에서 보듯이 해당 프로젝트를 실행하면 프로젝트에서 사용하는 프로세스 모델 정보가 프로세스 역할(Role)에 따라 단계, 부단계에서 수행해야할 활동 정보가 표시되고, 해당 활동에서 필요로 하는 산출물과 작성해야할 산출물이 표시된다. 뿐만 아니라, 활동에 사용되는 도구에 대한 정보도 뜨게 된다. 맨 아래쪽에는 활동의 세부 활동인 스텝(Step)이 상세히 기술된다. 만약, 해당 도구가 로컬 환경에서 구동이 가능하다면 Tool에 표시된 항목을 더블클릭할 경우 관련 소프트웨어가 실행된다.

5.2 모델링 도구

항해 모델과 컴포넌트 대화 모델을 위한 모델링 도구를 ArgoUML[22]에 기반하여 개발하였다. ArgoUML은 UML을 지원하기 위한 오픈소스 프로젝트(OpenSource Project)의 일환으로 개발된 UML 지원 도구이다. 자바 환경에서 개발되었으며 UML의 각종 다이어그램을 제작, 저장할 수 있으며 확장이 가능하도록 모듈 구조로 개발되어 있다.

그림 10에서 보듯이 XML 형식으로 미리 정의된 항해 모델을 읽어 들여 항해 다이어그램을 화면에 표시할 수 있으며 XML 형태로 추출이 가능하다. 또한 PageView, VirtualPageView등과 같은 페이지 뷰를 추가하고 페이지뷰 사이의 Navigation 연결도 추가할 수 있는 편집 기능이 있다. 모델 뷰 상단의 아이콘과 드래그를 통해 쉽게 할 수 있도록 GUI를 제공한다.

그림 11에서 보듯이 컴포넌트 대화 모델링 도구의 다

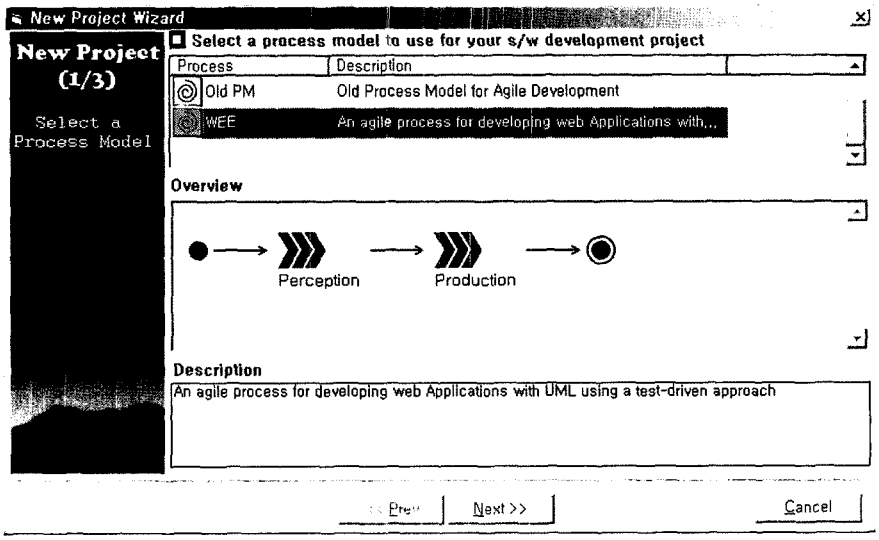

그림 8 프로젝트에서 적용할 프로세스 선택

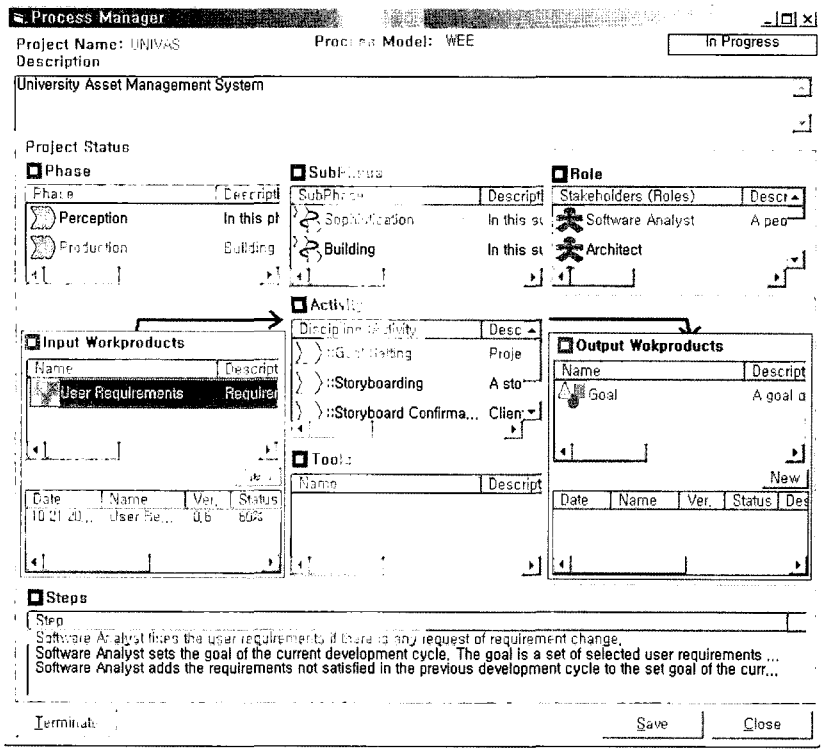

그림 9 새로운 요구사항 산출물이 추가된 화면

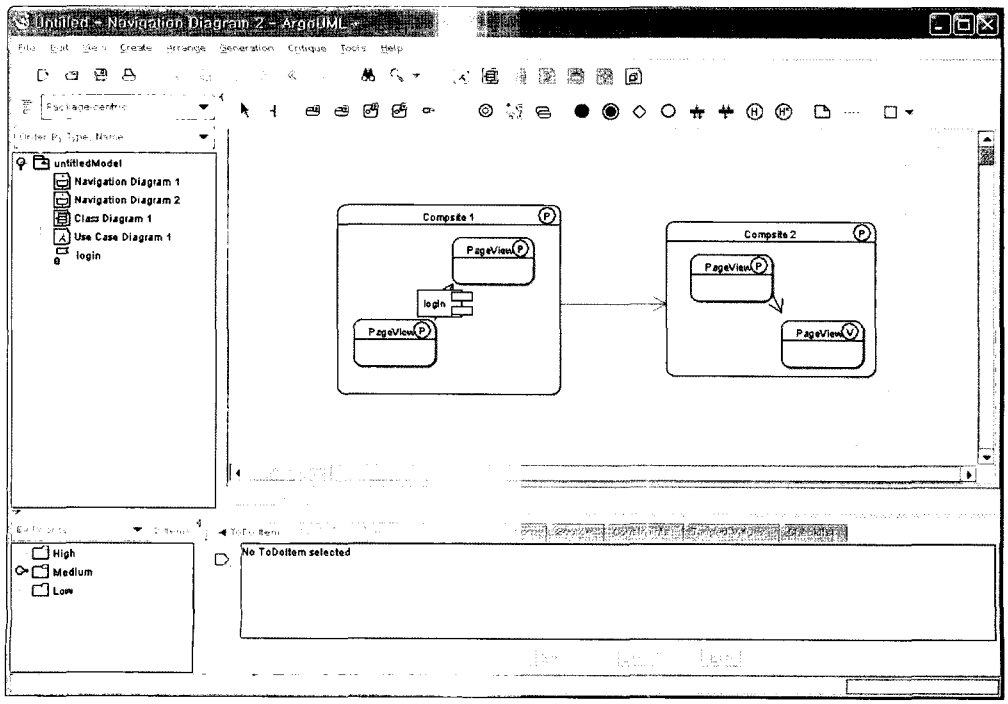

그림 10 항해 모델 도구

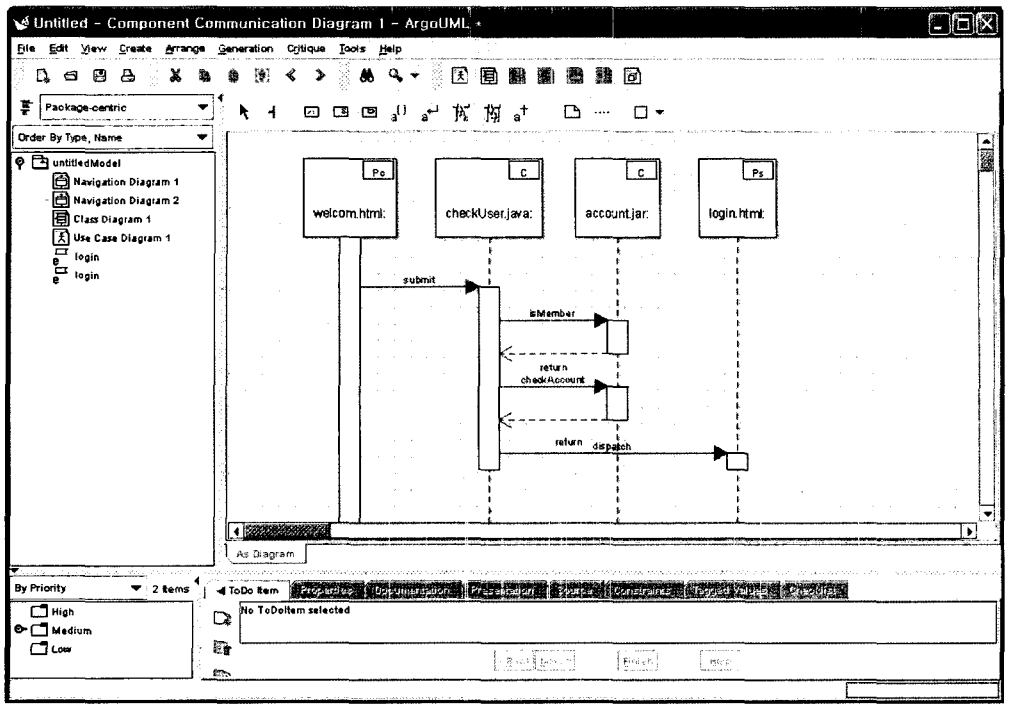

그림 11 완성된 컴포넌트 대화 모델

이어그램 영역 상단에는 다이어그램 편집 도구가 있다. 이 도구를 이용하여 ClientPage, ServerPage, Component를 추가하고 Call Action, Return Action 연결하는 것이 가능하다. 컴포넌트 대화 모델의 기본적인 편집은 기존의 시퀀스 다이어그램과 크게 다르지 않지만, 흐름이 ClientPage에서 시작되어 ServerPage에서 종료된다는 점이 다르다.

다이어그램 메뉴의 ClientPage 추가 아이콘을 선택한 후 다이어그램 영역에 클릭하면 ClientPage를 추가할 수 있다. 추가된 ClientPage의 이름을 적절히 변경시키는 것도 가능하다. ServerPage를 추가하고자 할 때는 다이어그램 메뉴의 ServerPage 추가 아이콘을 선택한 후 다이어그램 영역에 클릭하면 ServerPage를 추가할 수 있다. Component를 추가하기 위해서는 다이어그램 메뉴의 Component 추가 아이콘을 선택한 후 다이어그램 영역에 클릭하면 된다. 모든 Lifeline이 추가되었으면 이제 그들 간의 호출 관계를 표시할 수 있다. 다이어그램 메뉴의 Call Action 아이콘이나 Return Action 아이콘을 이용하여 각각의 호출 관계를 연결한다.

본 연구에서 개발한 도구가 각 모델링 활동과 연계되어 실행된다. 페이지 모델은 스테레오 타입을 이용한 UML 클래스 다이어그램으로 모델링 되므로 기존 UML모델링 도구가 실행된다.

6. 사례 연구

사례 연구로 논문이나 이전에 개발된 여러 컴포넌트, 공유 파일 등의 다양한 형태의 자료를 관리하기 위한 '학내 자료 관리 시스템(University Asset Management System: 이하 UNIVAS)'을 개발하였다. 익명 사용자는 UNIVAS를 검색하여 필요한 자료를 다운로드 받을 수 있으며 회원은 자료를 업로드하고 관리할 수 있다. UNIVAS는 몇 단계의 개발 주기를 통해 구축되었다. 설계 단계가 가장 선행되었으며 몇 번의 점진적인 작업을 통해 UNIVAS의 각 모듈들이 구현되었다. 프로세스 매니저를 이용해 전체 개발을 관리하였으며 모델링 도구를 이용해 모델링 작업을 수행하였다.

첫 번째 개발 주기에서는 논문 관리와 회원 관리를 위한 모듈을 개발하였다. 이 장에서는 두 번째 개발 주기에 해당하는 컴포넌트 관리 모듈 개발 주기를 사례로 들 것이다. 이 모듈의 몇몇 기능들은 첫 단계에서 개발된 회원 관리 기능들과 거의 유사하며 공유할 수 있다. 하지만 업그레이드 기능은 오직 이 개발 주기에서만 사용되는 기능이며 사례연구에서는 이 기능에 대해 중점적으로 기술하였다.

업그레이드 기능에 대한 사용자 요구 사항의 일부가 표 3에 나타나 있다. 오류 처리에 대한 특별한 요구 사항이 없기 때문에 오류가 발생시에는 오류 페이지로 오

표 3 사용자 요구 사항 기술서

유즈케이스 UC2: 컴포넌트의 업그레이드 **범위:** UNIVAS 컴포넌트 관리 모듈 **레벨:** 사용자 **주 액터:** 회원(인증을 거친 사용자) **주 시나리오 (주 흐름):** ... 2. 회원은 필드를 모두 채운다. (* 표시가 된 항목은 반드시 포함되어야 한다.) 입력이 완료되었으며 '파일 찾기' 버튼을 눌러 업그레이드할 파일을 선택한 후 '전송' 버튼을 눌러 업그레이드를 요청한다. 3. 업그레이드가 정상적으로 완료되었으면 'Successfully Upgraded'라는 메시지를 포함한 결과페이지가 제공된다.

류 메시지를 전달하는 것으로 처리하기로 한다. 그림 12는 사용자 요구 사항으로부터 페이지 모델링을 활동을 통해 생성된 모델에서 업그레이드 부분만 상세히 표현한 것이다.

프로세스 매니저는 각 프로세스 역할에게 필요한 산출물이 준비되었는지 여부를 검사하며 프로젝트 참여자들은 자신들의 활동에 대한 간단한 설명을 볼 수 있다. 인지 단계에서 요구사항 정리를 위해 필요한 문제 기술서와 페이지 모델이 완료된 상태이고, 소프트웨어 분석가는 해당 산출물을 더블 클릭 함으로써 프로젝트 서버에 보관되어 있는 산출물을 확인할 수 있다. 이러한 입력 산출물을 이용하여 요구사항을 작성하게 된다.

작성된 산출물을 프로젝트 서버에 업로드 하는 과정에서 요구사항을 필요로 하는 다른 역할들의 입력 산출물로써 자동적으로 추가된다. 비슷한 방식으로 나머지 작업도 필요한 산출물을 이용해 계속해서 새로운 산출물을 생성하게 된다. 프로젝트를 진행하면서 얻게 되는 모든 실제 산출물들은 프로세스 매니저를 통해 접근, 수정, 추가 등이 가능하다.

소프트웨어 분석가는 제작 단계의 초기에 인지 단계에서 작성된 사용자 요구 사항을 기반으로 하여 현재 개발 주기의 목표를 설정한다. 다음으로 소프트웨어 분석가는 스토리보드를 작성한다. 그림 13은 업그레이드 기능에 대한 html 폼을 포함하는 스토리보드의 일부이다. 스토리보드에 의하면 이 페이지는 입력 필드들이 표시되어 있으며 사용자의 액션에 대한 반응 방법을 서술하고 있다. 스토리보드의 작성이 완료된 후 소프트웨어 분석가는 스토리보드를 중심으로 항해 모델링 도구를 이용하여 항해 모델을 작성한다.

그림 14는 작성된 항해 모델이다. 이 항해 모델을 보면 몇 가지 문제가 될 수 있는 항해를 발견할 수 있는데, 만약 업그레이드 도중 오류가 발생하면 사용자는 index.html로 다시 돌아가서 처음부터 작업을 반복해야

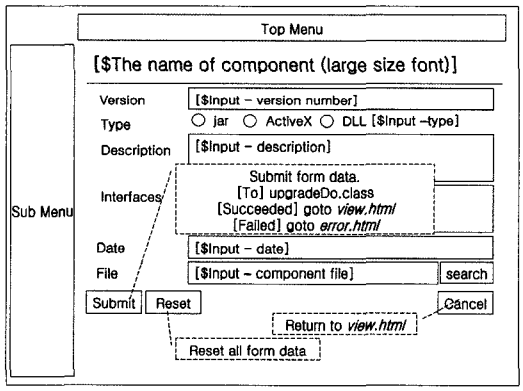

그림 13 스토리보드의 일부

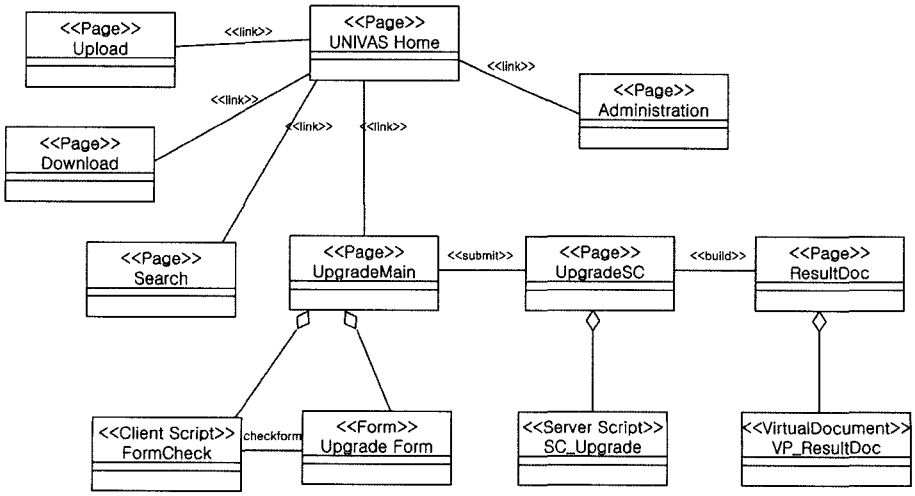

그림 12 UNIVAS 페이지 모델

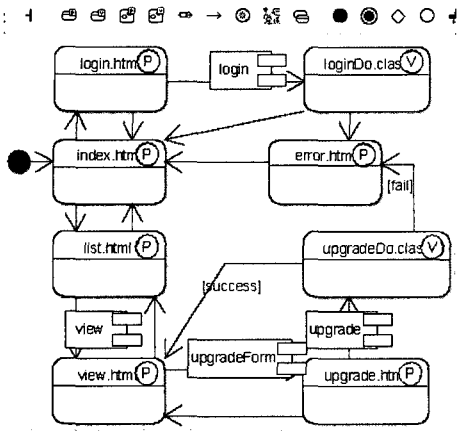

그림 14 항해 모델

만 하는데 사용자의 입장에서 굉장히 번거로운 작업이 될 수 있다. 이는 업그레이드 기능에 대한 오류 처리를 고려하지 않고 upgradeDo.class에서 error.html로 오류 메시지를 보내기 때문에 발생하는 문제이다.

이의 해결을 위해 항해 모델을 다음과 같이 수정했다. "만약 업그레이드 도중 오류가 발생한다면 다시 upgrade.html 페이지로 돌아간 후 사용자로 하여금 재시도가 가능하도록 한다." 그림 15는 그림 14의 항해 모델에서 수정된 부분을 보여준다. 변경된 사항은 설정된 목표에도 반영을 하여야 하며 표 4는 오류 처리가 포함된 수정된 목표의 일부이다.

스토리보드의 수정이 완료된 후 UI 개발자는 스토리보드를 기반으로 UI 프로토타입을 작성한다. 이렇게 작성된 UI 프로토타입은 화면의 구성과 항해에 대한 고객의 확인을 받는데 사용된다.

그와 동시에 컴포넌트 개발자는 항해 모델과 재사용 컴포넌트, 컴포넌트 리스트를 이용해 컴포넌트 대화 모델을 작성한다. 그림 16은 그림 14에 나타난 'upgrade' Submit에 대한 컴포넌트 대화 모델이다. 이 모델을 이용해 각 컴포넌트들의 인터페이스를 확인할 수 있으며 이 인터페이스는 컴포넌트 개발 과정에 반영된다.

실질적인 개발은 구축 부단계에서 이루어진다. 각 컴포넌트 개발자는 compupcation.jar파일과 upgradeDo.class 등을 포함한 필요한 컴포넌트들을 개발하였으며 시험 사례를 거치도록 하여 모든 오류를 검증한 뒤 두 번째 개발 주기를 완료한다. 그림 17은 이렇게 완성된 업그레이드 기능에 대한 페이지의 화면이다.

본 연구에서 제안하는 모델링 방법과 프로세스를 다

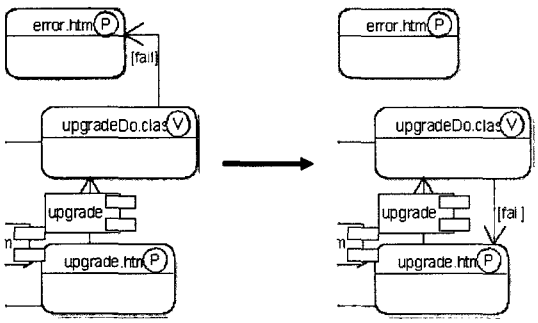

그림 15 수정된 항해 모델

표 4 오류 처리가 포함된 목표

...
만약 회원이 '전송' * 표시가 된 모든 필드를 입력하지 않고 버튼을 눌렀다면 업그레이드 페이지를 다시 사용자에게 제공하며 모든 입력 값은 그대로 유지가 되어야 한다. 또한 상단에 '[필드명] field should not be blank.' 메시지를 보여주도록 한다.

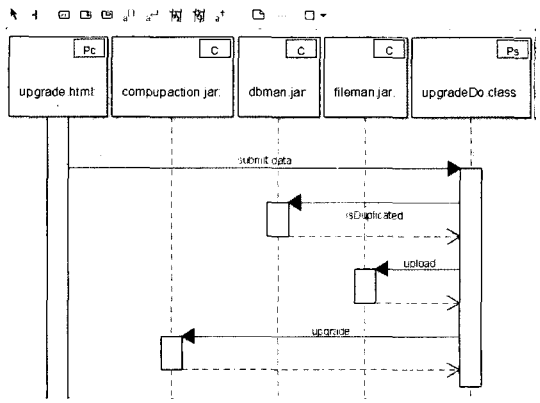

그림 16 컴포넌트 대화 모델

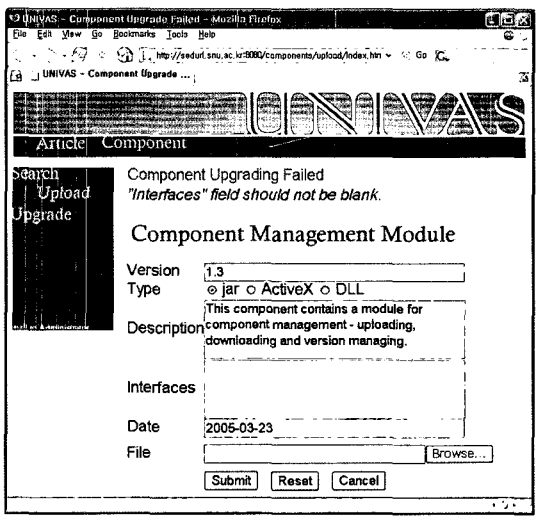

그림 17 개발된 웹 애플리케이션의 화면

른 연구와 비교 분석한다. 우선, 웹 응용 모델링 방법을 RMM, OOHDM 그리고 WebML과 비교 분석하였다 (표 5). 웹 응용 모델링 방법은 웹 응용 그 자체를 잘 표현해야 함은 물론이고 개발의 측면도 고려해야 한다. 이러한 점을 고려하여 모델, 지원도구, 동적 페이지의 고려, 사용자와의 상호작용, 사용자 적응성의 측면에서 비교 분석하였다.

대부분의 모델링 방법들이 공통적으로 개념 모델과 항해 모델을 사용한다. 그리고 RMM과 OOHDM, WebML의 경우 표현 모델을 사용하는 반면 제안한 모델은 표현 모델 대신에 스토리 보드와 UI 프로토타입을 사용하여 표현 계층의 모델링을 한다. 다른 모델링 방법과 달리 제안한 모델링 방법만의 특징적인 모델은 컴포넌트 대화 모델이다. 이는 스토리 보드, 항해 모델과 상호 관계를 맺으면서 사용된다. 그리고 모든 모델링 방법들이 고유한 방법을 적용한 도구를 사용한다. RMM에서는 RMCase, OOHDM은 OOHDMWeb 그리고 WebML은 WebRatio Site development studio 24라는 도구의 지원을 받는다. 제안한 모델링 방법은 기본적으로 UML을 확장한 모델을 사용하므로 UML을 지원하는 도구는 모두 사용이 가능하며, 본 연구에서 개발한 항해 모델링 도구와 컴포넌트 대화 모델링 도구가 사용가능하다.

대용량 정보를 실행 중에 생성하는 동적 페이지는 복잡한 웹 응용에서 중요한 요소이다. 제안한 모델링 방법을 제외하고는 동적 페이지에 대한 고려가 없다. 제안한 모델링 방법에서는 항해모델과 컴포넌트 대화 모델을 사용하여 동적 페이지를 생성하는 컴포넌트를 모델링 할 수 있다.

RMM을 제외한 나머지 모델링 방법들은 사용자의 상호작용을 모델링 할 수 있다. OOHDM과 WebML은 표현 모델을 사용하여 사용자가 웹 응용을 사용한다. 하지만 이 모델은 표현 계층의 인터페이스를 정의하지만 인터페이스의 내용과 위치 정도만 정의 모델링 한다. 반면에 본 연구에서 제안한 모델링 방법에서는 표현 계층을 스토리 보드와 UI 프로토타입을 사용하여 사용자 인터페이스를 직접적으로 표현하는 것뿐 아니라 표현 계층 전체의 테마와 컬러 등을 비롯한 보다 직관적이고 정확한 모델링이 가능하다.

WebML을 제외한 다른 모델링 방법들이 사용자 적응성을 고려하지 않고 있다. WebML의 경우 사용자 모델과 적응 모델을 사용하여 사용자를 모델링 하고 개별 사용자에 대한 적응성을 모델링 한다. 이 부분은 제안한 모델링 방법에서 지원이 약한 부분이다.

최근의 연구로 OOHDM의 확장격인 SHDM이 있는데 도메인 관련 언어(Domain Specific Language)를 기존 모델 기반 방법론에 조합하여 웹 애플리케이션의 빠른 프로토타이핑을 가능하게 한다는 이점을 가진다. 빠른 웹 애플리케이션 개발을 목표로 한다는 점에서 본 개발 방법론과는 유사하지만, SHDM의 경우에는 도메인 관련 언어를 이용하고, 본 연구에서는 기민한 프로세스를 채용한다는 점이 차이점이다. 또한 UWE의 경우에는 UML 확장 메타모델을 사용한다는 점과 ArgoUML을 확장한 ArgoUWE라는 모델링 도구를 지원한다는 점에서는 본 연구와 매우 유사하지만 전체적인 프로세스와 통합된 개발 환경을 제시하지 못한다는 점에서는 차이점을 가진다.

두 번째로 본 연구에서 제안한 프로세스를 RUP(Rational Unified Process)[23], XP, SCRUM 그리고 AWE와 비교 분석하였다. 웹 응용 개발 프로세스는 웹 응용의 특성을 반영하면서 동시에 웹 산업의 특징도 반영하여야 한다. 본 연구에서 제안한 개발 프로세스를 다른 소프트웨어 개발 프로세스와 비교하는 기준은 적응성, 짧은 생명주기, 시각적 모델, 프로젝트 규모, 인적 요소이다.

표 6은 위의 기준으로 평가한 결과이다. 비교의 대상이 되는 모든 개발 프로세스가 요구사항 변경에 대한 적응성을 가진다. 하지만 본 연구에서 제안하는 개발 프로세스에서는 스토리 보드와 UI 프로토타입을 초기에 개발하여 이를 통해 고객과의 의사소통을 원활히 하여

표 5 웹 응용 모델링 방법의 비교

	제안한 모델	RMM	OOHDM	WebML
모델	페이지 모델, 항해 모델, 컴포넌트 대화 모델	개념 모델, 행해 모델, 표현 모델	요구사항 분석 모델, 개념 모델, 항해 모델, 표현 모델	개념 모델, 항해 모델, 표현 모델, 사용자 모델, 적응 모델
지원도구	항해 모델링 도구 컴포넌트 대화 모델링 도구	RMCase	OOHDMWeb	WebRatio Site development studio 24
동적 페이지	CCM	고려 없음	고려 없음	고려 없음
사용자 상호작용	고려함	고려 없음	고려함	고려함
사용자 적응성	고려 없음	고려 없음	고려 없음	고려함

표 6 웹 응용 개발 프로세스의 비교

	제안한 프로세스	RUP	XP	SCRUM	AWE
적응성	O	O	O	O	O
짧은 생명 주기	O	X	O	O	O
시각적 모델	UML 확장	UML	X	X	WAE
프로젝트 규모	소~중	대	소	없음	없음
인적 요소	협업 고려, 능력 요구, 자가 관리	협업 고려	협업 고려, 능력 요구, 자가 관리	협업 고려, 능력 요구	협업 고려, 능력 요구

요구사항의 적응성을 실현한다. RUP와 같은 프로세스를 제외한 나머지 개발 프로세스는 짧은 개발주기를 지원하는 프로세스이다. 물론 Agile RUP와 같이 경량화된 프로세스도 있다. 본 연구에서 제안한 프로세스에서는 각 개발 주기의 초기에 해당 개발 주기의 목표를 설정한다. 이 목표를 적절하게 정의하여 전체 개발이 점진적이고 반복적으로 이루어지게 한다. RUP에서는 주로 UML을 많이 사용하고 AWE에서는 WAE라는 모델을 사용한다. 본 연구에서 제안하는 프로세스에서는 UML을 확장한 모델을 선택적으로 사용한다. UML의 모든 모델을 사용하는 것이 아니고 웹 응용 개발에 반드시 필요한 모델만을 사용한다. 그리고 추가적으로 UML에는 없는 스토리 보드와 UI 프로토타입을 사용한다.

프로젝트의 규모 면에서 RUP는 대규모 프로젝트에 적합하고 XP는 소규모 프로젝트에 적합하다. 본 연구의 프로세스는 소규모 또는 중규모 프로젝트에 적합하다. 본 연구의 프로세스가 개발자를 비롯한 모든 참여자의 협업을 요구하기 때문이다. 반면에 SCRUM과 AWE는 프로젝트 규모에 대한 고려가 없다. 인적 요소를 고려해 보면 RUP는 협업 고려 요소만을 가지고 본 연구의 프로세스와 XP만이 협업 고려, 능력 요구, 자가 관리 요소를 가진다. 나머지 프로세스는 자가 관리 요소가 없다.

본 연구에서 제안한 개발 프로세스는 UML에 기반한 최소한의 산출물을 정의하고, 시험 기반 구현을 채택함으로써 모델 기반 및 시험 기반 접근법의 장점을 모두 가진다. 그리고 웹 응용에만 초점을 맞추어 범용적인 다른 프로세스 보다 웹 응용 개발에 적합하다. 그리고 개발 과정 내의 일부 활동들이 병렬적으로 이루어짐으로 개발 인력을 효율적으로 사용하고 빠른 개발을 가능하게 한다.

7. 결 론

본 논문에서는 확장 UML 모델에 기반하여 요구사항의 변경에 기민하게 대응할 수 있는 웹 애플리케이션 개발 프로세스를 제안하였다. 또한 이 프로세스를 실행하고 최적화할 수 있는 프로세스 지원 도구를 통해 프로세스를 최적화하여 좀더 실제적이고 유용한 프로세스로 개선하였다. 웹 애플리케이션 모델링을 위한 지원도구를 개발하고 실제 사례 연구를 통해 유용성을 검증하였다.

본 논문에서 제안한 개발 프로세스는 UML에 기반한 최소한의 산출물을 정의하였다. 웹 애플리케이션에만 초점을 맞추어 범용적인 다른 프로세스 보다 웹 애플리케이션 개발에 적합하다. 그리고 개발 과정 내의 일부 활동들이 병렬적으로 이루어짐으로 개발 인력을 효율적으로 사용하고 빠른 개발을 가능하게 한다.

본 논문에서 제안하는 웹 공학 프로세스는 기본적으로 개발 참여자들이 하나의 팀을 구성하는 것을 전제로 한다. 그러므로 기업형 웹 애플리케이션(enterprise Web application)의 경우와 같이 많은 개발 인력이 투입되며 다양한 팀이 동시에 개발에 참여하는 개발 프로젝트에는 한계를 갖는다. 본 연구에서 제안하는 웹 애플리케이션 개발 방법론의 유용성을 검증하기 위해 사례 연구가 진행되었으나 웹 애플리케이션의 도메인 및 프로젝트 크기 등이 한정되었다. 보다 다양한 사례 연구를 통한 검증이 요구된다.

참 고 문 헌

[1] A. Ginige and S. Murugesan, "Web Engineering: An Introduction," *IEEE Multimedia*, Vol. 8, Iss. 1, pp. 14-18, 2001.

[2] T. Isakowitz, E. A. Stohr and P. Balasubramanian, "RMM: a Methodology for Structured Hypermedia Design," *Communications of ACM*, Vol. 38, No. 8, pp. 34-44, 1995.

[3] F. Frasincar, G. J. Houben and R. Vdovjak, "An RMM-Based Methodology for Hypermedia Presentation Design," ADBIS 2001, *LNCS 2151*, pp. 323-337, 2001.

[4] S. Ceri, P Fraternali and A. Bongio, "Web Modeling Language (WebML): modeling language for designing Web sites," *Proc. of WWW9*, 2000.

[5] WebML: The Web Modeling Language, http://www.webml.org

[6] D. Schwabe, G. Rossi and S. Barbosa, "Systematic Hypermedia Application Design with OOHDM," *Proc. of ACM-Hypertext96*, 1996.

[7] D. Schwabe and G. Rossi, "An Object Oriented Approach to Web-Based Application Design," *Theory and Practice of Object Systems*, Vol. 4, No.4, 1998.
[8] L. Baresi, F. Garzotto and P. Paolini, "Extending UML for Modelling Web Applications," *Proc. of HICSS-34*, 2001.
[9] O. D. Troyer and C. Leune, "WSDM: A User-Centered Design Method for Web Sites," *Proc. of WWW07*, pp. 85-94, 1998.
[10] K. Schwaber and M. Beedle, *Agile Software Development with SCRUM, Prentice-Hall*, 2001.
[11] A. McDonald and R. Welland, "Agile Web Engineering (AWE) Process," Technical Report, University of Glasgow, Scotland, 2001.
[12] A. McDonald and R. Welland, "Evaluation of Commercial Web Engineering Processes," ICWE 2004, *LNCS 3140*, pp. 166-170, 2004.
[13] A. McDonald and R. Welland, "Agile Web Engineering (AWE) Process: Multidisciplinary Stakeholders and Team Communication," ICWE 2003, *LNCS 2722*, pp. 515-518, 2003.
[14] D. A. Nunes and D. Schwabe, "Rapid Prototyping of Web Applications combining Domain Specific Languages and Model Driven Design," *ICWE 2006*, 2006.
[15] N. Koch and A. Kraus, "Towards a Common Metamodel for the Development of Web Applications," ICWE 2003, *LNCS 2722*, pp. 497-506, July 2003.
[16] S. Slemi, N. Kraiem, and H. Ghezala, "Toward a Comprehension View of Web Engineering," ICWE 2005, *LNCS 3579*, pp.19-29, 2005.
[17] K. Beck, *Extreme Programming Explained: Embrace Change*, Addison-Wesley, 2001.
[18] Rational Process Workbench, http://www-128.ibm.com/developerworks/rational/library/4144.html#N10104
[19] OMG, Software Process Engineering Metamodel (SPEM), Ver. 1.1, OMG, 2005.
[20] R. S. Pressman, Software Engineering: A Practitioner's Approach, McGraw-Hill 6th ed., 2004.
[21] K. Beck, Test Driven Development: By Example, Addison Wesley, 2002.
[22] ArgoUML, http://argouml.tigris.org/
[23] P. Kruchten, The Rational Unified Process: An Introduction, Addison-Wesley Professional 2nd ed., 2000.

이 기 열

1999년 서울대학교 수학·계산·통계학부 졸업(학사). 2001년 서울대학교 대학원 전기·컴퓨터공학부 졸업(석사). 2001년~현재 서울대학교 대학원 컴퓨터공학부 박사과정. 관심분야는 웹 공학, 소프트웨어 프로세스, 테스팅

정 우 성

2003년 서울대학교 수학·계산·통계학부 졸업(학사). 2003년~현재 서울대학교 대학원 컴퓨터공학부 석박사 통합과정(박사과정). 관심분야는 웹 공학, 소프트웨어 테스팅, 적응형 소프트웨어

이 춘 우

2002년 서울대학교 컴퓨터공학부 졸업(학사). 2002년~현재 서울대학교 컴퓨터공학부 석박사 통합과정(박사과정) 관심분야는 웹 테스팅, 그리드 컴퓨팅

이 병 정

1990년 서울대학교 계산통계학과 졸업(학사). 1990년~1998년 현대전자 소프트웨어 연구소 주임연구원. 1998년 서울대학교 대학원 전산과학과 졸업(석사). 2002년 서울대학교 대학원 컴퓨터공학부 졸업(박사). 2002년~현재 서울시립대학교 컴퓨터과학부 조교수. 관심분야는 소프트웨어 품질 시험/평가/인증, 소프트웨어 재공학, 웹 공학

김 희 천

1989년 서울대학교 계산통계학과 졸업(학사). 1991년 서울대학교 대학원 전산과학과 졸업(석사). 1998년 서울대학교 대학원 전산과학과 졸업(박사). 1998년~2003년 한세대학교 컴퓨터공학과 조교수 2004년~현재 한국방송통신대학교 컴퓨터과학과 조교수. 관심분야는 소프트웨어공학

우 치 수

1965년~1972년 서울대학교 응용수학과 졸업(학사). 1972년~1974년 한국과학기술원 연구원. 1975년~1977년 서울대학교 대학원 전산과학과 졸업(석사). 1977년~1982년 서울대학교 대학원 전산과학과 졸업(박사). 1978년 영국 라퍼러 대학 연구원. 1975년~1982년 울산대학교 전자계산학과 조교수, 부교수. 1985년~1986년 미국 미시간대학교 postdoc. 1988년~1990년 한국정보과학회 총무 이사. 1990년~1992년 한국정보과학회 학회지 편집위원장. 1992년~1993년 한국정보과학회 부회장. 1996년~1998년 한국정보과학회 소프트웨어공학연구회 운영위원장. 1982년~현재 서울대학교 컴퓨터공학부 교수