웹 어플리케이션 개발을 위한 사용자 인터페이스 패턴 모델 개발과 효용성에 관한 연구

A Study on Development and Efficiency of UI Design Pattern Model for Web Application

김창겸, ChangGyum Kim*, 유승연, SeongYeon Yu*, 정영웅, YoungWoong Jeong**
*LG CNS S/W Innovation Center, **홍익대학교 영상대학원

요약 최근 국내외 소프트웨어 개발 특성의 변화인 잦은 요구사항의 변경, 단축된 비즈니스 라이프사이클, 비용절감의 압력등과 맞물려 생산성을 높이기 위한 개발 접근방식이 요구되었으며, 소프트웨어의 재사용 및 높은 생산성을 제공할 수 있는 대표적인 방법이 디자인 패턴이다. 본 연구는 웹 어플리케이션 시스템 개발에 실제 적용가능한 UI 디자인 패턴모델을 개발하기 위해 20 개의 웹 어플리케이션 시스템을 표본으로 추출하여 1000 여개의 단위업무화면을 분석하였다. 개발한 어플리케이션 UI 디자인 패턴 모델을 실무 전문가(UI 디자이너, 개발자)들에게 이해도, 사용적합성, 생산성측면에서 검증하여 실무에 적용가능한 모델임을 객관적으로 증명하였다.

핵심어: 패턴, 재사용, 사용자 인터페이스, 인터렉션 디자인, 웹 어플리케이션, CRUD

1. 서론

SI(System Integration)프로젝트에서 갈수록 고객의 요구가 복잡해지고 다양한 기술을 필요로 함에 따라 최근 웹 어플리케이션 시스템(Web Application System)에 대한 개발 생산성은 갈수록 중요시 되고 있다. 더욱이 시스템에서 사용자가 직접 조작하는 UI에 대한 요구사항은 개발이 끝나는 시점까지 끊임 없이 변경되고 추가되어 프로젝트의 생산성 저하에 큰 영향을 주는 요인이 되고 있다. 이러한 문제들은 수많은 프로젝트에서 비슷한 유형으로 반복적으로 발생하고 있어 이에 대한 효율성 향상이 요구되고 있다.

이러한 문제를 해결하기 위해서는 성공적인 디자인 해결책에 대한 재사용이 필요하며 성공적인 해결책을 재사용하기 위해서는 이를 효과적으로 정리하고 전달하기 위한 체계가 있어야 하는데 그것이 바로 패턴이다.

1990년대 중반부터 웹 시스템, 객체 지향 시스템 구축이 본격화되면서 개발 패러다임의 전환을 외치는 목소리가 강해졌으며 이 시기부터 패턴에 대한 중요성이 대두되기 시작했다. 어려운 객체지향적 사고를 돕고 새로운 디자인 개념들을 습득하기 위해 패턴은 중요한 수단으로 인식되어 왔다 그러나 국내외적으로 UI분야의 디자인 패턴에 대한 연구는 초기 상태로 주로 실무를 위한 경우보다는 개념적 이론을 정립하는 내용이 주를 이루고 있다.

따라서 본 연구의 목적은 웹 어플리케이션 개발 프로젝트 현장에서 실제 적용될 수 있으며, 생산성을 높일 수 있는 UI 디자인 패턴 모델을 개발하여 제시하는 것이다.

2. 이론적 배경

2.1 웹 어플리케이션의 이해

웹 어플리케이션은 하이퍼미디어라는 형식을 통해 기존의 소프트웨어 어플리케이션을 통합한 형태이다. 이로 인해 웹 어플리케이션은 단순한 정보 획득이라는 목적뿐만 아니라 정보의 전달, 수정, 삭제등의 복잡한 양상을 띠고 있다[9]. Conallen[6]은 웹 어플리케이션은 서버에 비즈니스 로직을 가지고 있어 사용자의 입력이 비즈니스 로직의 상태에 영향을 미치는 시스템이라고 정의하여 검색 엔진이나 홍보성 홈페이지는 웹 어플리케이션에 포함되지 않는다고 하였으며 이와 비슷한 정의로 Gellersen과 Gaedke[8]는 웹 어플리케이션은 일정한 비즈니스 로직을 가지고 웹에 의존하여 실행되는 소프트웨어 어플리케이션이라고 하였다. 즉, 웹 어플리케이션은 웹 시스템을 확장하거나 그 위에 비즈니스 기능을 추가한 것으로 업무처리를 주목적으로 하는 업무처리 시스템이라고 할 수 있다. 이러한 업무처리의 기본단위는 데이터를 입력(Create)하고 조회(Read)하고 수정(Update)하고 삭제(Delete)하는 일련의 정보 처리 프로세스이며 이러한 작업은 페이지 단위로 이루어지며 사용자 인터페이스에 직접적인 영

향을 준다.

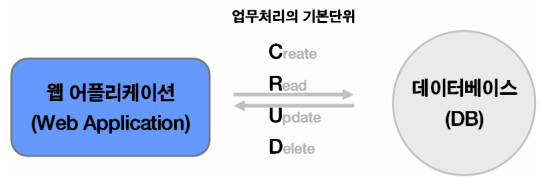


그림 1 웹 어플리케이션의 업무처리 프로세스

2.2 패턴의 이해

패턴 개념을 처음 제시한 Alexander[4]의 기본 아이디어는 건축물의 설계에 빈번하게 발생하는 동일 설계 내용이 있으며, 이런 것들을 하나의 패턴으로 보고 다른 건축물 설계에 재사용하면 여러 가지 면에서 이득을 가져 다 준다는 것이다.

많은 사람들이 패턴에 대한 기원을 알렉산더가 1977년에 출간한 저서인 'A Pattern Language'로 알고 있으나 실제로는 1968년에 출간한 저서인 'A Pattern Language Which Generates Multi-Service Center'에서 처음으로 '패턴'이라는 개념을 사용하였으며 이를 건축설계과정의 중요한 개념으로 제시하였다. 이 저서에는 다목적 회관(Multi-Service Center) 설계를 위한 64개의 패턴을 제공하고 있다. 또한 이 패턴들의 네트워크(Network)가 제공되고 있는데 이 네트워크란 '패턴들이 어떻게 서로 맞물리게 되는 가를 보여주는 시스템'이 된다. Alexander[4]는 이 패턴 시스템을 언어와의 비유로 설명하고 있다. 한 사람이 자신의 생각을 표현하고자 할 때, 그는 자신이 가지고 있는 어휘 중에서 표현하고자 하는 생각에 맞는 단어들을 선택하고, 이것들을 현재 사용하고 있는 언어의 문법에 따라 배열 한다. 마찬가지로 한 건축가가 건물을 설계할 때도 그는 우선 그가 작업하고 있는 특정 건물에 맞는 패턴들을 이미 제공된 원형 건물 디자인(Prototype Building Design) 즉, 패턴 시스템에 나열된 패턴의 집합에서 선택을 한다.

여기서 설계문제의 해결에 대한 알렉산더의 접근방법을 다시 살펴보면, 건축가는 마치 우리의 언어에 미리 어휘가 주어졌듯이 미리 만들어진 패턴의 집합을 검색하여 그의 특정 설계 프로젝트의 상황(Context)에 맞는 패턴을 선택하게 된다. 선택이 끝나면 그는 이 패턴들을 조합하는데 이 작업은 언어의 문법과 비슷한 구조 혹은 시스템에 따라 큰 것에서 작은 것으로의 순서로 각각의 패턴이 서로 들어 맞도록 하는 것이다[3].

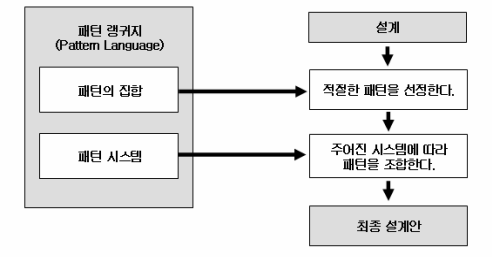


그림 2 패턴 랭귀지를 이용한 건축 설계과정

Alexander[5]는 추후에 패턴 랭귀지(A Pattern Language)라는 저서를 출간하면서 패턴에 대한 총정리를 하여 총 230여 개의 패턴을 담았으며 각각의 패턴은 우리를 둘러싸고 있는 환경(Context)에서 반복적으로 나타나는 특정한 문제(Problem)와 그에 대한 해결책(Solution)을 설명하고 있으며 그 해결책은 계속 사용될 수 있기 때문에 동일한 과정을 반복할 필요가 없다고 하였다. 여기서 언급된 상황(Context), 문제(Problem), 해결책(Solution)을 패턴의 3요소라고 하며, 패턴을 설명하고 기술하는 데 있어서 핵심이 되는 요소이다. 이를 알렉산더 양식(Alexanderian Form)이라고도 한다.

표 1은 패턴의 3요소에 대한 설명이다.

표 1 패턴의 3요소

알렉산더 양식	설명
상 황(Context)	패턴이 적용되어질 수 있는 상황, 문제가 발생하는 상황
문 제(Problem)	패턴이 해결하고자 하는 문제, 주어진 상황에서 발생하는 요구사항
해결책(Solution)	발생한 문제에 대한 해결방안, 재사용이 가능한 공통된 해결책

패턴에 대한 알렉산더의 기본 아이디어는 건축 영역을 벗어난 소프트웨어와 같은 다른 영역에서도 적용할 수 있었는데 1987년 워드 커닝험(Ward Cunningham)과 켄트 벡(Kent Beck)은 알렉산더의 아이디어를 사용해서 객체지향 프로그래밍에 대한 다섯 가지의 패턴을 만들었다. 그리고 이 내용은 " Using Pattern Languages for Object-Oriented Programs"라는 제목으로 정리되어 객체 지향에 관한 세계적인 컨퍼런스(conference)인 OOPSLA-87(Object-Oriented Programming, Systems, Languages & Applications 87)에 논문으로 발표되어 디자인 패턴에 대해서 학계에 공식적으로 알리는 계기가 되었다. 여기서 디자인 패턴이란 여러가지 문제에 대한 설계 사례를 분석해서 서로 비슷한 문제를 해결하기 위한 설계를 분류하고, 각 문제 유별로 가장 적합한 설계를 일반화시켜 패턴으로 정립한 것을 의미한다.

1990년대 초에는 Gamma, Helm, Vlissides 및 Johnson[8]은 객체지향 소프트웨어 설계에 가장 영향을 주었던 " Design Patterns: Elements of Reusable Object-Oriented Software"이란 책을 저술하였다. 이 책은 1994년에 발표되어 디자인 패턴에 대한 아이디어를 소프트웨어 분야에 널리 알리는 계기를 만들었으며, 이 책은 책의 제목이 긴 이유로 인해서 'GoF(Gang of Four: 4명의 저자를 가리킴) Design Pattern'라는 별칭으로 불리고 있다.

GoF의 디자인 패턴은 총 23개로 특정업무 영역과는 상관없이 일반적인 소프트웨어 설계에서 발행할 수 있는 기본적인 문제들과 객체지향 특성을 최대한 활용한 해결책을 제시하고 있으며 크게 생성 패턴(Creational Pattern)과 구조 패턴(Structural Pattern)그리고 행위 패턴(Behavioral Pattern)으로 분류하고 있다. 여기서 주목할 것은 GoF가 설명하고 있는 패턴들은 그들 스스로가 창안해 낸 것이 아니라, 단지

수많은 프로젝트에서 반복해 사용하는 디자인들을 파악하여 이를 모아 체계적으로 분류하고 문서로 정리한 것이라는 점이다. 즉 그들은 새로운 것을 발명한 것이 아니라 기존에 있던 것을 발견하여 정리를 한 것이며, 이때 패턴 개념을 적용한 것이다.

3. 선행 연구 분석

UI디자인 패턴은 소프트웨어 개발에 도입된 패턴의 개념이 다시 HCI분야에 도입된 것으로, 인터렉션 디자인(Interaction Design) 패턴, HCI패턴 등으로 불려지기도 하며 이 분야에서 활발한 연구활동을 하고 있는 Welie[12]는 인터랙션 디자인에서 어떻게 패턴 언어를 의미있고 실질적인 방법으로 구조화할 수 있는 지에 대한 방안으로서 하향식(top-down) 방식으로 상위 단계에서 하위 단계의 디자인 문제들까지를 계층적으로 구조화하는 접근법을 제시하였다.

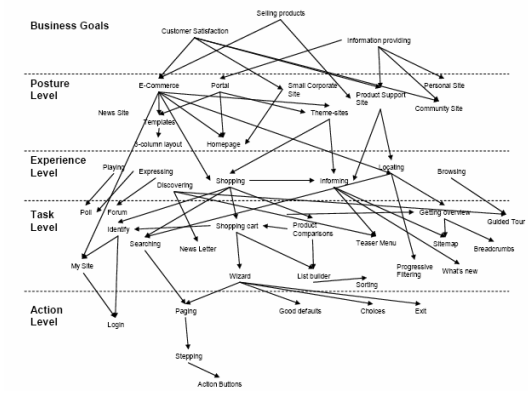


그림 3 Welie의 UI디자인 패턴 분류

그는 패턴을 사이트 유형, 경험, 태스크, 액션으로 그룹핑하였으며, 점점 구체화되며 상호 연관성이 있는 패턴 분류 체계를 제시하고 있다.

1) 사이트 유형 구역(Posture Type Level): 비즈니스에 따라 실질적으로 같은 목표를 갖고 구조적으로 유사한 사이트 컨셉을 갖는 많은 사이트들이 있는데, 이러한 사이트 유형을 분류하였다.

2) 경험 구역(Experience Level): 사용자가 사이트 컨셉 하에서 사용자의 주요 목적에 달성하고 주요 태스크를 수행하며 만족을 얻는 방법과 과정에 대해 분류하였다.

3) 태스크 구역(Task Level): 경험 패턴에서 보다 구체화된 패턴으로서, 사용자가 태스크를 수행하는 데에 있어 발생하는 일련의 인터랙션을 분류하였다.

4) 액션 구역(Action Level): 실제 태스크에 대해서 의미가 있는 세부적인 인터랙션을 분류하였다.

또한 그가 제시하는 패턴 양식은 “Problem”, “Use when”, “Solution”, “Why”, “More Examples”등으로 알렉산더 양식을 기반으로 작성되었으며 이러한 양식이 적용된 인터랙션 패턴에 대한 예[11]는 그림 4와 같다.

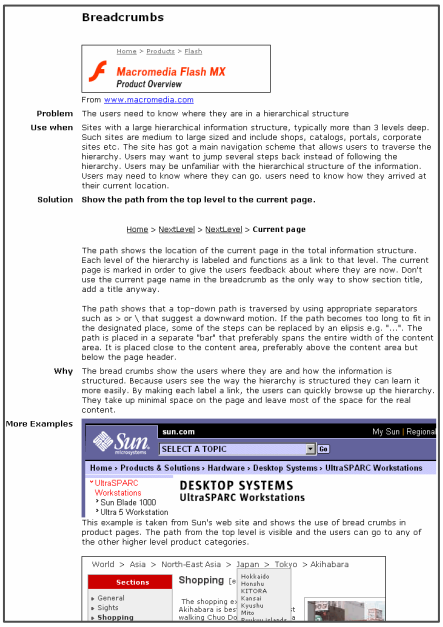


그림 4 Welie의 UI디자인 패턴에서 Breadcrumbs 패턴

국내에서 처음으로 UI패턴 연구를 수행한 최영완[2]은 그의 패턴 연구를 ‘인터랙션 디자인 지식화 프레임’으로 표현하였으며, 지식 관리를 중심으로 인터랙션 디자인 재사용을 위한 디자인 패턴을 제안 하고 있다. 그는 지식화 양식으로 Alexander[5]가 제시한 디자인 패턴의 기본 개념인 정황과 문제, 그리고 해결책의 요소 외에 인터랙션 디자인에 적합한 요소들로 인터랙션 디자인 패턴 양식을 구성 및 분류하였다.

그는 패턴 분류시 전문가 인터뷰 결과에 따라 디자인 문제의 유형을 분류하고 그에 따라 인터랙션 디자인 패턴을 태스크, 사용자, 기술, 메뉴 선택, 내비게이션, 입력, 표시, 가이드/피드백, 레이아웃, 아이콘/컬러 패턴으로 분류하였다.

1) 태스크 패턴 : 사용자가 수행해야 할 태스크 상의 문제에 대한 해결책을 다룬다. 사용자의 태스크 분석을 통해서 어떤 기능 또는 어떤 컨텐츠를 제품에서 제공해야 하는지 도출된다. 태스크는 태스크 성격에 따라서 다시 서브 태스크로 구성될 수 있으므로 태스크 간의 위계 관계가 형성될 수 있다.

2) 사용자 패턴 : 초보자, 중급자, 숙련자 등의 사용 숙련도 및 아이나 노인과 같은 나이에 따라서 발생하는 문제들을 다룬다. 인터렉션 디자인을 하기 위해서는 제품을 사용하는 사용자의 도메인 지식 여부나 컴퓨터 사용 경험이나 빈도 또는 사용 연령 등의 차이를 고려해야 한다.

3) 기술적 패턴 : 휴대폰, PDA, 디지털 TV, 윈도우 환경이나 웹 환경 등의 플랫폼 또는 디바이스 그리고 기술적인 상황 때문에 발생하는 문제들을 다룬다. 데스크탑에 비해서 상대적으로 작은 화면 크기를 가진 휴대폰에서 발생하는 문제나 데스트탑의 마우스와 TV의 리모콘과 같이 디바이스의 변화에 따라서 발생하는 인터렉션 디자인에 관련된 문제를 다룬다.

4) 메뉴 선택 패턴 : 메뉴 디자인과 관련된 문제를 다룬다. 풀다운 메뉴, 팝업 메뉴, 탭 메뉴 등이 메뉴 선택 패턴에 포함된다.

5) 네비게이션 패턴 : 화면안 또는 화면간의 네비게이션 또는 메뉴안에서의 네비게이션등의 문제를 다룬다. 페이지나 화면을 순차적으로 이동할 수 있는 위저드나 사용자의 포커스를 강제적으로 한 윈도우에 두게 하는 모달 윈도우 등이 이에 속한다.

6) 입력 패턴 : 시스템이 사용자부터 데이터를 입력 받는 것과 관련된 문제를 다룬다. 사용자가 입력 항목에 대해서 직접 키보드로 입력하거나 선택하는 건 뿐 아니라 양식 작성을 포함한 문제를 다룬다. 날짜 입력 패턴이나 한 목록의 항목을 다른 목록으로 이동시켜 입력하는 부분 선택 패턴 등이 이에 속한다.

7) 표시 패턴 : 표시 패턴은 데이터 뿐 아니라 문서 등을 표시하는 것과 관련된 문제를 다룬다. 목록 보기에서 항목들을 구분시켜 보이게 하기 위해서 색깔을 지정하는 칼라 레이블 등이 이에 속한다.

8) 가이드 및 피드백 패턴 : 사용자에게 오류 메시나 도움말 제공과 관련된 문제를 다룬다. 회원가입과 같은 입력 화면에서 필수 입력 항목 표시 패턴 등이 이에 속한다.

9) 레이아웃 패턴 : 화면이나 웹 페이지의 그리드나 배치와 관련된 문제를 다룬다. 그리드 패턴, 페이지 고정 넓이 패턴과 페이지 가변 넓이 패턴이 이에 속한다.

10) 아이콘 및 컬러 패턴 : 아이콘, 이미지, 칼라, 타이포그라피 등의 비주얼 디자인 관련 문제를 다룬다. 칼라 코드 영역등이 이에 속한다.

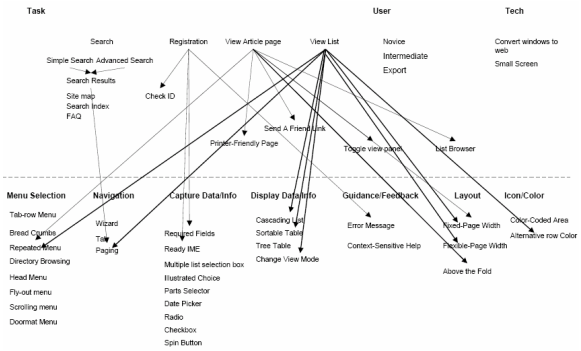


그림 5 최영완의 인터렉션 디자인 패턴 분류

최영완[2]의 패턴 양식의 특징은 증거 기반의 디자인 패턴을 위하여 해결책에 사용된 논리적인 이유 외에 디자인 원칙, 관련 리서치 자료등의 항목을 알렉산더 양식에 추가를 한 점이다. 인터랙션 디자인 해결책에 대한 신뢰도나 타당성을 증명하려면, 실험이나 사용자 테스트와 같은 리서치가 필요하고 해결책에 적용된 논리적 이유나 디자인 원칙들이 필요한데 알렉산더의 양식에는 해결책의 증거로 연구 결과나 디자인 원칙, 논리적 이유 등이 따로 구분되어 있지 않기 때문이다.

또한 알렉산더의 양식은 패턴의 신뢰도를 신뢰도가 있는 것과 진행중인 것, 그리고 신뢰도가 확인되지 않는 3단계로 구분하고 있는데, 최영완[2]은해결책의 증거 정도에 따라서 신뢰도를 구분하는 것이 필요하다고 생각하였으며. 이러한 양식이 적용된 인터랙션 패턴에 대한 예는 그림 6와 같다.

그림 6 최영완의 인터렉션 디자인 패턴에서 Tab row menu 패턴

이상과 같이 GoF의 디자인 패턴이나 Welie, 최영완의 디자인 패턴이나 모두 저마다 각영역에서 중요한 문제들을 해결하기 위한 솔루션을 제공하고 있다. GoF는 객체지향 소프트웨어 설계에서 발생하는 문제에 대하여 개발자들이 겪는 문제에 대한 해결책을 제공하여 주고 있으며, Welie와 최영완은 UI디자인에서 발생하는 문제에 대하여 디자이너들이 겪는 문제에 대한 해결책을 제공하고 있다.

이러한 패턴은 재사용이 가능한 UI디자인에 대한 해결책을 일관적이며 가장 지식을 효과적으로 축적하고 전달하기 위한 표준적인 포맷으로 제공하는 것이라고 말할 수 있으며 [9], 패턴은 훌륭한 디자인 해결책에 대한 지식 축적 및 전달 체계라고 할 수 있다. 그러나 기존의 UI디자인 패턴은 디자이너의 관점에 지식을 축적하고 문제점과 해결책을 제시하였으며, 사용성 원칙이나 디자인 가이드라인이 해결책의 중요한 기준으로 사용되며, 웹사이트의 UI를 중심으로 되어 있음을 알 수 있다. 따라서 개발자 중심적이고 업무처리 중심적인 웹 어플리케이션 개발을 위한 새로운 UI디자인 패턴의 필요성이 제기되었다.

4. 어플리케이션 UI디자인 패턴모델 개발

4.1 패턴 도출을 위한 화면유형 수집 및 분석

웹 어플리케이션 시스템 구축은 고객의 비즈니스를 어떠한 형태로 수행하고 처리하는지를 정확히 파악해야 하는 과제를 안게 된다. 웹 어플리케이션 시스템의 구축에서 발생하는 대부분의 문제들은 개발자들이 이러한 사용자의 작업 패턴과 선호도를 파악하지 못하는데서 발생하는 경우가 많다.

더욱이 시스템에서 사용자가 직접 조작하는 UI에 대한 요구사항은 개발이 끝나는 시점까지 끊임 없이 변경되고 추가되어 프로젝트의 생산성 저하에 큰 영향을 주는 요인이 되고 있다. 이러한 문제들은 수많은 프로젝트에서 비슷한 유형으로 반복적으로 발생하고 있어 본 패턴 연구를 통해서 그 해결책을 제시하고자 한다.

앞서 GoF의 디자인 패턴에서 언급한 바와 같이 패턴이라는 것은 창안되는 것이 아니라 수많은 프로젝트에서 반복해 사용하는 디자인들을 파악하여 이를 모아 체계적으로 분류하고 문서로 정리한 것이라고 하였다. 즉 패턴화는 무질서 속에서 잠재되어 있는 일련의 규칙을 발견해내는 작업이라고 할 수 있다. 따라서 이러한 원리에 입각해 웹 어플리케이션의 UI디자인 패턴을 도출하기 위하여 국내 대표적인 SI업체인 LG CNS에서 2001년도에서 2005년도까지 구축한 국내의 웹 어플리케이션 시스템 중 프로젝트 관리시스템에 등록되어 있는 총 226개의 웹기반 프로젝트 프로젝트를 조사 범위로 잡았다. 그리고 공공, 금융, 전자, 제조, 물류, 건설등 비즈니스 도메인별로 20개 프로젝트를 표본으로 추출하였다.

분석한 단위업무 수는 1,018개로 화면수로 따지면 3000개정도이며, 분석기간은 총 3개월이 소요되었으며, 투입인원은 풀타임으로 시니어급 UI전문가 2명, 파트타임으로 시니어급 SW아키텍트 1명이 투입되었다. 분석대상은 웹 어플리케이션 업무화면에서 한번에 처리하는 Data의 양과 CRUD처리방법, 그리고 각각의 단위업무를 수행하기 위해 일어나는 화면 네비게이션 플로우간의 상관관계이다.

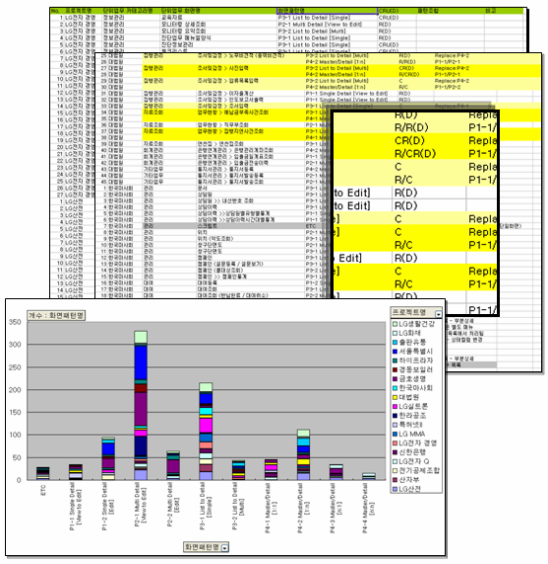


그림 7 웹 어플리케이션 UI분석 데이터

분석내용을 정리하면 첫째, CRUD와 업무화면간의 상관관계이다. 같은 유형의 업무를 처리하는 데 있어서 여러가지 방식을 사용한다는 것이다. 예를 들어 매출전표관리라는 단위업무가 있는데 매출전표를 등록 하는 업무화면이 있다. 근데 A라는 시스템은 신규 매출전표를 등록하는 업무화면을 개발하였을 때 1건씩 입력하게 하였고, B라는 시스템은 같은 업무화면을 개발하였을 때 여러 건을 한화면에서 입력하게 하였다. 그리고 C라는 시스템은 우선 매출전표를 등록하기 전에 목록을 조회하고 나서 목록에 비어있는 입력 필드에 값을 입력하고 콤보박스에서 값을 선택하도록 하였다.

이처럼 개발자가 웹 어플리케이션을 개발 할 때, 도대체 이 업무를 멀티건으로 처리하는 화면으로 하는게 좋은지 단건식 처리하는 화면으로 처리하는 게 좋은지, 그리고 어떠한 방식으로 화면 네비게이션 플로우를 가져가야 좋을지 많은 고민을 하게 되고, 여러가지 선택의 기로에 서서 불충분한 근거를 가지고 제각각의 선택을 하게된다.

둘째. 이러한 문제는 서로 다른 시스템뿐만 아니라 한 시스템의 비슷한 유형의 업무를 개발하는데로 발생하고 있어서 비효율적이며 일관성없는 복잡한 UI를 양산한다는 것이다. 그 원인을 파악해보면 업무처리방식에 따른 UI형태를 프로젝트에서 패턴으로 정의하지 않고 개발자들이 그때 그때 고객의 요구사항에 따라 일관성 없이 개발을 하였기 때문이다. 규칙성 없는 UI는 사용성(Usability)이 떨어지게 되는데 그 중에서도 학습성(Learnability)과 기억용이성(Memorability)이 많이 떨어지게 된다. 이는 곧 업무생산성(Efficiency) 저하로 이어져 고객의 불만 요인이 되고 프로젝트의 품질저하 요인이 된다.

4.2 패턴 모델의 개발 과정

데이터를 수집/분석하고 여기에 특정 기준을 통해 패턴을 도출하고 패턴들 간에 유기적 관계를 가진 하나의 패턴 모델이 개발되기까지의 과정을 간단하게 정리하면 그림 8과 같다.

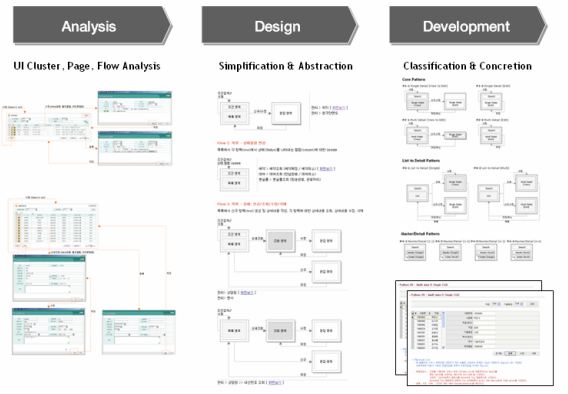


그림 8 패턴 모델의 개발과정

분석(Analysis)단계에서는 웹 어플리케이션 시스템의 UI를 화면영역, 화면스타일, 업무흐름에 따라 각 유형별로 분류를 하고, 디자인(Design)단계에서는 분석하고 분류한 유형에 따라 불필요한 군더더기를 제거하는 단계로 단순화 및 추상화를 하게된다. 구현(Development)단계에서는 패턴을 모델링하고 각 패턴별로 구체화된 프로토타입을 개발하게 된다.

패턴을 도출하는 데 있어서 가장 중요한 것은 패턴화하려는 것이 자주 발생하는 일반적인 문제여야 한다는 것이다. 특별한 경우에 가끔 발생하는 문제는 패턴화 할 수 있는 문제가 아니며, 그러한 것을 단지 패턴 양식에 따라 기술하였다고 해서 패턴이 되는 것은 아니다. 그것은 단지 하나의 유형에 대한 가이드일 뿐이다. 즉 범용성이 떨어짐으로 패턴이라고 할 수가 없다.

또한 새로운 패턴을 도출하고 제시함에 있어서 독창적인

아이디어를 통한 해결책보다는 오히려 널리 알려져 있는 문제에 대한 표준적인 해결책이 좋은 패턴으로서 인정을 받는다[1].

4.3 어플리케이션 UI디자인 패턴 모델

본 연구를 통해서 제시하는 모델인 어플리케이션 UI디자인 패턴 모델은 어플리케이션의 기본적인 데이터 입출력 방식인 CRUD를 기반으로 하는 화면의 Flow를 포함하고 있으며 비전문가라도 그 패턴의 구조와 작동방식을 직관적으로 이해할 수 있도록 실체화된 프로토타입이 연계되어 있다.

어플리케이션 UI디자인 패턴 모델에는 총 10개의 패턴이 정의되어 있으며 크게 Core 패턴과 List to Detail 패턴, Master/Detail 패턴 3가지로 분류할 수 있다.

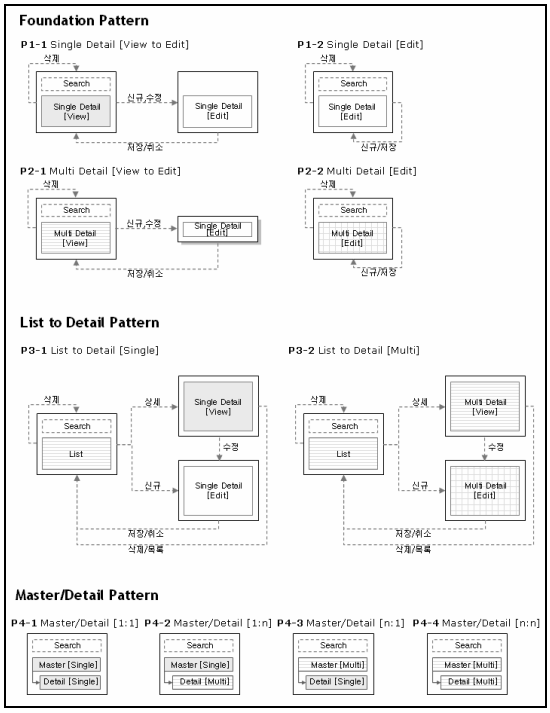


그림 9 어플리케이션 UI디자인 패턴 모델

1)기본(Foundation) 패턴 : 1건 처리와 멀티(n)건 처리의 기본적인 형태를 가지고 있는 패턴으로 패턴 응용과 조합의 근간이 되는 패턴이다.

2)리스트 디테일(List to Detail) 패턴 : 멀티건의 상세에 대한 CRUD를 처리하는 패턴으로서 초기 멀티건 List(목록)의 메인 화면에서 Detail(상세)에 대한 CRUD가 처리되는 서브화면으로 분기되는 화면 구조로 이루어진다.

3)마스터 디테일(Master/Detail) 패턴 : 상위 데이타(Master)와 종속 관계에 있는 하위 데이터(Detail)에 대한 CRUD를 한 화면에서 처리하는 패턴이다. 마스터 디테일 패턴은 고밀도의 한 화면에서 상위와 하위의 데이터에 대한 CRUD를 쉽게 처리할 수 있다.

어플리케이션 UI디자인 패턴 모델에서 가장 기본이 되는 모듈단위는 UI Cluster인데 하나의 화면을 구성하는 사용자 인터페이스 요소들을 기능단위로 묶은 것이다. 모듈화된 UI 요소들 간의 조합 및 관계를 통해서 하나의 구조 체계를 형성하고 복잡한 기능과 많은 양의 데이터를 한 화면에서 처리해야 하는 경우가 프로젝트에서 발생하게 되면, 패턴끼리 조합을 하여 복잡한 업무를 수행하는 또하나의 패턴을 정의하여 해당 프로젝트의 표준 패턴으로 사용할 수 있다.

표 2 패턴 모델의 기본 구성 단위인 UI Cluster

UI Cluster	Diagram	Description
Search영역	Search	•Data를 조회하기 위한 조건을 입력할 수 있도록 구성된 영역. •조건영역으로 조회 또는 편집영역의 Data를 조회하기 위한 조건을 선택하거나 입력한다. •Search영역에 올 수 있는 UI Control은 콤보박스, 인풋박스, 라디오 버튼 등이다.
Single Detail [View]영역	Single Detail [View]	•1건의 Data를 조회할 수 있도록 구성된 영역. •일반적으로, 여러 항목 별로 해당 Label과 Data가 일대일로 매칭되어 보여진다.
Single Detail [Edit] 영역	Single Detail [Edit]	•1건의 Data를 편집할 수 있도록 구성된 영역. •입력/선택 가능한 UI Control가 포함되는 것을 기본으로 하되, 버튼 액션 등에 의해 업데이트 편집이 발생하는 경우도 포함한다.
Multi Detail [View]영역	Multi Detail [View]	•n건의 Data를 조회할 수 있도록 구성된 영역. •일반적으로, 여러 항목 별로 해당 Label과 n건의 Data가 일대다로 매칭되어 보여진다. •List to Detail 패턴에서 List 영역으로 사용된다.
Multi Detail [Edit]영역	Multi Detail [Edit]	•n건의 Data를 편집할 수 있도록 구성된 영역. •입력/선택 가능한 UI Control가 포함되는 것을 기본으로 하되, 버튼 액션 등에 의해 업데이트 편집이 발생하는 경우도 포함한다.
List 영역	List	•List to Detail 패턴에 사용되는 Multi Detail [View] 영역
Master [Single]영역	Master [Single]	•Master/Detail 화면의 Master 부분으로서 Single Detail 패턴이 올 수 있다.
Detail [Single]영역	Detail [Single]	•Master/Detail 화면의 Detail 부분으로서 Single Detail 패턴이 올 수 있다.
Master [Multi]영역	Master [Multi]	•Master/Detail 화면의 Master 부분으로서 Multi Detail 패턴과 List to Detail 패턴이 올 수 있다.
Master [Detail]영역	Detail [Multi]	•Master/Detail 화면의 Detail 부분으로서 Multi Detail 패턴과 List to Detail 패턴이 올 수 있다.

4.4 어플리케이션 UI디자인 패턴 양식

어플리케이션 UI디자인 패턴 모델은 Alexander[5]가 제안한 패턴의 기본 양식을 따르면서 시스템 개발에 직접 도움을 줄 수 있도록 보완을 하였다. 패턴 양식에 대한 상세한 설명은 표 3과 같다.

표 3 어플리케이션 UI디자인 패턴 모델의 패턴 양식

Term	Description
Problem	• 해당 패턴을 사용해야 하는 핵심적인 문제나 니즈(사용자 요구사항)를 정의한다. • 해당 패턴을 사용하려면 설계/개발하고자 하는 화면이 1차적으로 Problem에 부합하여야 한다.
Use When	• 해당 패턴에 대해 주로 발생할 수 있는 구체적인 정황을 기술한다. • 기본 Problem 하에서 해당 패턴을 사용할 때 실질적으로 필요한 세부적인 상황 또는 사용자의 업무처리 형태를 다루며, 자주 사용되는 CRUD의 조합, 비즈니스 케이스 등에 대해 기술할 수 있다. • '기본업무로 수행'이라 함은 해당업무를 메뉴명으로 하는 경우를 말한다.(xx조회, xx등록, xx수정…)
Solution	• Context와 Requirement를 충족시키는 화면패턴을 CRUD 처리 및 화면전환 등의 관점에서 기술한다.
Pattern Diagram	• 해당 패턴의 구조 및 CRUD 작동 방식을 한눈에 알아보기 쉽도록 개념적으로 추상화한 다이어그램이다.
Pattern Structure	• 해당 패턴의 내비게이션 구조와 화면 구조에 대해 설명하며, 화면의 각 영역에서 이뤄지는 작업에 대해 기술한다.
Event	• 해당 패턴을 구현시 발생하는 Action 또는 Navigation 버튼에 대한 이벤트를 Client Event, Server Event, Navigation Event로 나누어 기술한다.
Example	• 해당 패턴이 사용될 수 있는 실제 사례를 보여줌으로써 패턴에 대한 이해를 돕는다.
Sample Source	• 해당 패턴을 구현하는 샘플 프로그램에 대한 소스로 연결된다.
Developer's Guide	• 해당 패턴을 구현하는 샘플 프로그램을 개발하는 데에 필요한 개발용 가이드로 연결된다.
Related Pattern	• 해당 패턴과 관련이 있거나 조합 가능한 패턴들을 서술한다.

어플리케이션 UI디자인 패턴 모델에서 제시하고 있는 10개의 패턴 중 가장 기본이 되는 'Single Detail[View to Edit]' 패턴은 그림 10과 같으며 1건의 업무를 처리하는 사용자 인터페이스를 디자인할 때의 문제와 해결방안을 다루고 있다. 추상화된 패턴 다이어그램을 통해 UI설계에 대한 컨셉

을 잡을 수가 있으며 예제화면으로 실제 구현시에 어떻게 응용할 수 있는지를 가늠해 볼 수가 있다. 각 영역에서 어떤 이벤트가 발생하는 지도 기술되어 있으며 개발 소스와 가이드도 함께 제공된다.

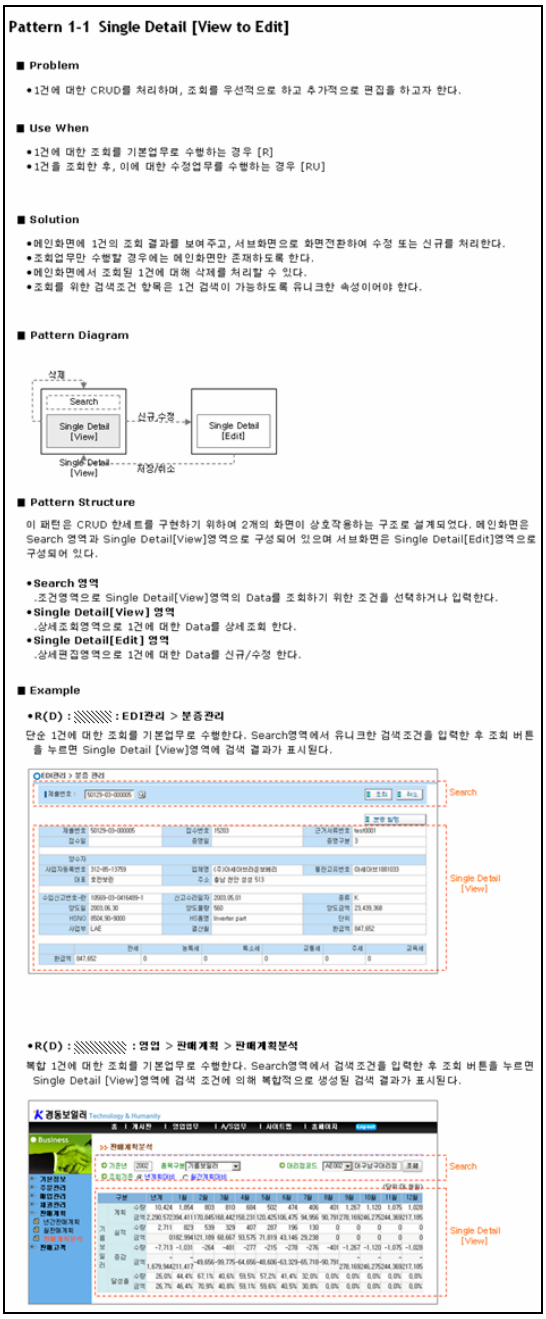


그림 10 어플리케이션 UI디자인 패턴 모델의 Single Detail 패턴

5. 실무 전문가 집단에 의한 객관성 검증

본 연구에서 제시된 어플리케이션 UI디자인 패턴 모델에 대한 객관성 검증을 위하여 관련 전문가 그룹을 대상으로 설문조사를 실시하였다. 설문항목은 이해도측면 2문항, 사용 적합성 측면 3문항, 생산성 효과측면 4문항으로 라이커트 척도(Likert scale) 방식의 7점 척도를 사용하였고, 1점은'전혀 동의하지 않는다'로 7점은'매우 동의 한다'로 설정하여 측정하였다.

UI디자이너 그룹(UI, GUI, Usability, IA등의 전문가)과 시스템 개발자 그룹(Modeler, Architect, SE등의 전문가)으로 나누었으며, 시스템 개발자 그룹의 경우 대부분 L사의 개발자들이며 본 연구에서 제시한 패턴 모델을 이미 알고 있거나 프로젝트에서 활용해본 적이 있는 사람들이다. UI디자이너 그룹의 경우 한국HCI연구회와 HED(Human Experience Design) 리서치 그룹에서 활동하는 UI디자이너들로 본 연구에 대한 세미나를 2006년 5월과 6월에 실시하고 세미나 참석자를 대상으로 설문조사를 실시하였다.

수집된 설문지 중 불성실한 응답이나 분석 자료로서 부적절한 설문지 5부를 제외한 70부를 분석 자료로 이용하였으며 세미나 참석자 중에 개발자로 응답한 3명은 개발자 그룹에 포함시켜서 분석을 하였다. 수집된 설문자료는 SPSS 11.0 통계프로그램을 통하여 다음과 같이 분석하였다.

첫째, 조사대상자의 인구통계학적 특성과 제시된 UI패턴의 이해도와 사용 적합성, 그리고 생산성 효과 등의 전반적인 응답분포를 살펴보기 위해 빈도분석을 실시하였다.

둘째, 본 연구에서 제시한 UI패턴의 이해도와 사용 적합성, 그리고 생산성 효과등의 구성항목별 차이분석을 위하여 t-test와 일원변량분석(one-way ANOVA)을 실시하였으며, 일원변량분석 후 유의한 차이가 있는 경우 Duncan의 사후검정을 실시하였다.

셋째, 개발자와 UI디자이너의 제시된 UI패턴의 이해도와 사용 적합성, 그리고 생산성 효과 등의 인식에 대한 비교분석을 위하여 t-test를 실시하였다.

표 4 조사대상자의 일반적 특성

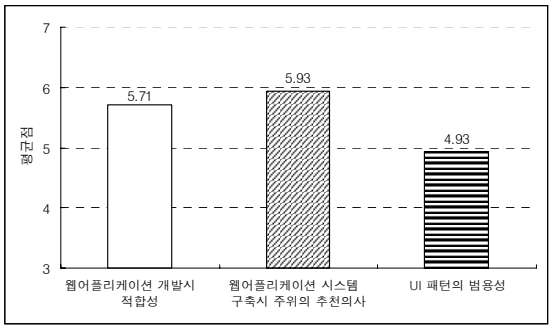
구 분		개발자		UI 디자이너		전 체	
		빈도(명)	백분률(%)	빈도(명)	백분률(%)	빈도(명)	백분률(%)
웹관련 업무경력	없음			4	(11.4)	4	(5.7)
	3년 미만	4	(11.4)	8	(22.9)	12	(17.1)
	3-5년 미만	5	(14.3)	6	(17.1)	11	(15.7)
	5-7년 미만	14	(40.0)	8	(22.9)	22	(31.4)
	7년 이상	12	(34.3)	9	(25.7)	21	(30.0)
UI 디자인/개발 경력	없음			2	(5.7)	3	(4.3)
	3년 미만	9	(25.7)	12	(34.3)	20	(28.6)
	3-5년 미만	7	(20.0)	7	(20.0)	14	(20.0)
	5-7년 미만	12	(34.3)	8	(22.9)	20	(28.6)
	7년 이상	7	(20.0)	6	(17.1)	13	(18.6)
합 계		35	(100.0)	35	(100.0)	70	(100.0)
*UI 디자인/개발 분야	웹사이트	7	(15.5)	21	(43.8)	28	(30.1)
	웹어플리케이션	34	(75.6)	10	(20.8)	44	(47.3)
	데스크탑	4	(8.9)	7	(14.6)	11	(11.8)
	모발일			8	(16.7)	8	(8.6)
	가전제품			1	(2.1)	1	(1.1)
	기타			1	(2.1)	1	(1.1)
합 계		45	(100.0)	48	(100.0)	93	(100.0)

*복수응답

설문조사에 참여한 개발자 35명과 UI디자이너 35명의 일반적 사항은 표 4와 같다. 먼저 개발자에 대하여 살펴보면, 업무경력은 5-7년 미만이 40.7%로 가장 많았으며, 7년 이상이 34.3%, 3년 미만이 11.4%로 나타났으며, UI디자인/개발 경력은 5-7년 미만이 34.3%로 가장 많았고, 3년 미만이 25.7%, 3-5년 미만과 7년 이상이 20.0%로 분포하였다. UI디자인/개발 분야는 75.6%의 대부분이 웹어플리케이션 이었으며, 웹사이트가 15.5% 데스크탑이 8.9%로 나타났다.

표 5 UI패턴의 사용 적합성 구성항목별 비교

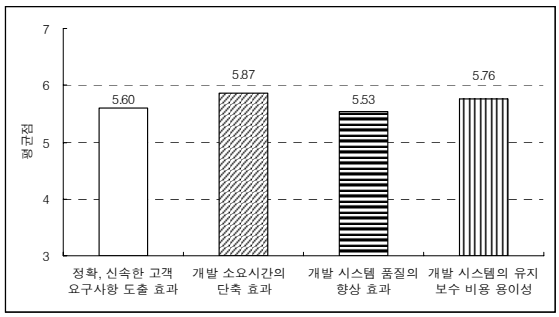
항 목	N	M	SD	F	p	Duncan' test
웹어플리케이션 개발시 적합성	70	5.71	1.01	14.901***	.000	A
웹어플리케이션 구축시 주위에 추천의사	70	5.93	1.23			B
UI 패턴의 범용성	70	4.93	1.17			B



설문 분석 결과 제시된 UI패턴의 사용 적합성은 평균 5.52로 비교적 높게 평가하였으며, 웹 어플리케이션 시스템 구축시 주위의 추천의사(5.93)와 웹어플리케이션 개발시 적합성(5.71)이 UI 패턴의 범용성(4.93)에 비하여 유의하게 높은 것으로 나타났다.

표 6 UI패턴의 생산성 효과 구성항목별 비교

항 목	N	M	SD	F	p	Duncan' test
정확하고 신속한 고객 요구사항의 도출 효과	70	5.60	1.18	1.385	.248	-
개발 소요시간의 단축 효과	70	5.87	1.05			
개발한 시스템의 품질 향상 효과	70	5.53	1.18			
개발한 시스템의 유지보수 용이성	70	5.76	.97			



설문 분석 결과 제시된 UI패턴의 생산성 효과는 5.69로 높게 평가하였으며, 통계적으로 유의한 차이는 나타나지 않았으나, 개발 소요시간의 단축 효과(5.87)가 가장 높게 평가하였으며, 다음으로 개발 시스템의 유지보수 비용 용이성(5.76), 정확하고 신속한 고객 요구사항의 도출 효과(5.60), 개발 시스템 품질의 향상 효과(5.53) 등의 순으로 높게 나타났다.

6. 결론 및 향후과제

이상과 같이 웹 어플리케이션 시스템의 복잡한 업무처리 형태를 단순화, 추상화(Abstraction)하여 CRUD중심의 10개의 UI디자인 패턴을 도출하고 체계화 하였으며 소프트웨어 공학의 객체지향 개념과 UI디자인 패턴을 접목하여 웹 어플리케이션 시스템 개발에 적용 가능한 모듈화, 구조화된 어플리케이션 UI디자인 패턴 모델을 제시하였다.

또한 제시된 패턴 모델이 웹 어플리케이션 개발에 적합하며, 프로젝트에 적용시 생산성 효과가 있는 모델임을 실무 전문가를 통해 객관적으로 입증을 하였다.

향후 연구에서는 실제 사용자 그룹을 대상으로 사용성 테스트를 실시하여 패턴 모델이 고객의 니즈에도 부합되었는지와 고객의 요구사항을 정확하게 도출하는 효과가 있는지에 대한 실증적인 연구가 필요하다.

참고문헌

[1] 신우창, 객체지향 소프트웨어 설계의 재사용을 위한 설계패턴 명세언어, 박사학위논문, 서울대학교 대학원, 2003.

[2] 최영완, 인터렉션 디자인 재사용을 위한 디자인 패턴에 관한 연구 : 지식 관리를 중심으로, 석사학위논문, 연세대 대학원, 2004.

[3] 최재필, 크리스토퍼 알렉산더의 여섯권의 저작에 나타난 건축설계과정의 개념의 변화, 대한건축학회 논문집, 12(2), 27-32, 1992.

[4] Alexander, C. A pattern language which generates multi-service centers. Berkeley, CA: Center for Environmental Struture, 1968.

[5] Alexander, C. A pattern language : Towns, buildings, construction, New York: Oxford University Press, 1977.

[6] Conallen, J. Building web application with UML, Boston, MA: Addison-Wesley, 1999.

[7] Fournier, R. A methodology for client/server and web application development, NJ : Yourdon Press, 1998.

[8] Gamma, E., Helm, R., Vlissides, J. and Johnson, R. Design patterns : Elements of reusable object-oriented software, Reading, MA: Addisson-Wesley, 1994.

[9] Gellersen, H. W., and Gaedke, M. “Object-Oriented Web Application Development”, IEEE Internet Computing, 3(1): 60-68, 1999.

[10] Tidwell, J. (1999). Common ground: A pattern language for human-computer interface design. http://www.mit.edu/~jtidwell/interaction_patterns.html

[11] Welie, M. "Patterns in interaction design", 2001. http://www.welie.com/patterns/index.html

[12] Welie, M., and Veer G.C. Pattern Languages in Interaction Design: Structure and Organization, In Proceedings of Interact ‘03, 1-5, September, Zürich, Switserland, Eds: Rauterberg, Menozzi, Wesson, pp527-534, IOS Press, Amsterdam, The Netherlands, 2003.