웹 기반 어플리케이션의 기능 테스트 자동화 방법

국 승 학[†] · 김 현 수[††]

요 약

최근 웹 어플리케이션은 급속도로 성장하였으며, 점점 더 복잡해지고 있다. 웹 어플리케이션이 복잡해질수록 품질에 관련된 다양한 요구사항이 증가하고 있다. 그러나 웹 어플리케이션의 테스트에 관한 연구 및 도구는 매우 부족하다. 이에 본 논문에서는 웹 어플리케이션에 대한 테스트 자동화 기법을 제안한다. 이를 위해 본 논문에서는 HTML 코드와 소스 코드로부터 분석 모델을 생성하고, 이 모델을 기반으로 테스트 대상을 파악하며, 테스트 케이스를 추출한다. 또한 테스트 드라이버와 테스트 데이터를 자동으로 생성하고, 그것들을 서버 내에 자동으로 배치함으로써 매우 쉽게 테스트 환경을 구축한다. 본 논문에서 제안하는 방법은 웹 어플리케이션에 대한 테스트의 전반적인 과정을 자동화하며, 이러한 자동화 방법은 기존의 연구에 비해 웹 어플리케이션의 기능 테스트의 효과를 높일 수 있다는 장점이 있다.

키워드 : 웹 기반 어플리케이션, 클라이언트 측 페이지, 서버 측 페이지, 소프트웨어 테스트, 테스트 자동화, 기능 테스트, 사용자 인터페이스 테스트

Automated Functionality Test Methods for Web-based Applications

Seung Hak Kuk · Hyeon Soo Kim

ABSTRACT

Recently web applications have grown rapidly and have become more and more complex. As web applications become more complex, there is a growing concern about their quality. But very little attentions are paid to web applications testing and there are scarce of the practical research efforts and tools. Thus, in this paper, we suggest the automated testing methods for web applications. For this, the methods generate an analysis model by analyzing the HTML codes and the source codes. Then test targets are identified and test cases are extracted from the analysis model. In addition, test drivers and test data are generated automatically, and then they are deployed on the web server to establish a testing environment. Through this process we can automate the testing processes for web applications, besides the automated methods makes our approach more effective than the existing research efforts.

Key Words : Web Based Applications, Client Side Pages, Server-Side Pages, Software Test, Test Automation, Function Test, User Interface Test

1. 서 론

최근 웹 어플리케이션은 급속도로 성장하였으며, 다양한 비즈니스 요구사항으로 인해 전통적인 어플리케이션은 웹 환경으로 빠르게 전환되고 있다[1]. 이러한 웹 어플리케이션은 JSP, Java Servlet, PHP, ASP, CGI, XML, ODBC, JDBC등과 같은 다양한 기술들을 적용한 동적인 컨텐츠(dynamic content)를 포함하고 있다. 또한 웹 어플리케이션을 프리젠테이션 계층과 비즈니스 계층, EIS(Enterprise Information System) 계층 등으로 나누는 다계층(multi-tier) 아키텍처가 적용되어 이에 대한 복잡도가 증가하고 있으며, 웹 어플리케이션에 대한 사용성, 신뢰성, 상호운영성, 보안과 같은 품질에 대한 요구사항이 증가하고 있다[2-6].

※이 논문은 2006년도 충남대학교 학술연구비의 지원에 의하여 연구되었음.
† 정 회 원 : 충남대학교 컴퓨터과학과 박사과정
†† 종신회원 : 충남대학교 전기정보통신공학부 부교수(교신저자)
논문접수 : 2007년 3월 20일, 심사완료 : 2007년 6월 15일

웹 어플리케이션에 대한 다양한 품질 요구사항을 만족하기 위해 어플리케이션 개발 과정에서 적절한 테스트 방법이 필수적이다. 각 개발된 페이지에서 기능상 오류가 없는지, 각 컴포넌트 사이에 병목 현상으로 인한 성능저하는 없는지, 많은 사용자의 동시 요청에 응답할 수 있는지, 사용자가 해당 어플리케이션을 사용하기 편리한지 등에 대한 다양한 테스트를 수행해야 품질 높은 어플리케이션을 개발할 수 있을 것이다. 이러한 웹 어플리케이션의 테스트는 그 목적에 맞는 방법 및 도구를 이용해야 한다. 성능 테스트를 위해서는 다양한 로드를 줄 수 있는 테스트 방법이 필요하며, 보안을 위해서는 보안 설정과 관련된 테스트 방법을 적용해야 한다. 특히 웹 어플리케이션의 기능을 테스트하기 위해서는 구성하는 각 페이지의 기능 및 연동되는 컴포넌트들 사이에서 기능이 정확하게 수행되는지 테스트할 수 있는 방법이 필요하다.

이러한 테스트에 대한 필요성은 많은 개발자, 사용자들이 인지하고 있지만, 이를 지원하는 자동화된 도구는 많지 않

다. 테스트 과정을 자동화한 도구는 대부분 가상의 사용자를 이용한 성능/부하 테스트에 관련된 도구들뿐이다. 물론 웹 어플리케이션의 기능 테스트에 관한 몇몇 연구들이 진행되어 있다. 하지만 이러한 연구들은 기존의 테스트 방법을 웹 어플리케이션에 적용하거나 테스트 모델을 개발하는 연구가 대부분이며, 실제 개발된 사례 및 자동화에 관련된 노력에 관한 것은 극히 드물다. 또한 현재 웹 어플리케이션 개발 현장에서 테스트에 많이 이용되는 몇몇 도구들이 존재하지만 이러한 도구들은 테스트 과정에서 많은 사용자의 개입을 요구한다.

웹 어플리케이션의 기능 테스트 자동화는 일반적인 어플리케이션의 테스트보다 복잡하다. 웹 어플리케이션은 다양한 페이지로 구성되고, 폼(form), 스크립트와 같은 내부 컴포넌트(inner component)를 포함하고 있어 테스트 자동화에 필요한 데이터를 추출하기가 쉽지 않다. 또한 웹 어플리케이션이 웹 브라우저를 통해서 사용자에게 보여지기 때문에 테스트 드라이버의 생성 및 결과 판단은 웹 브라우저를 기반으로 이루어져야 한다. 이러한 사실은 테스트 자동화를 어렵게 만든다. 현재 대부분의 웹 어플리케이션 테스트는 개발자 및 사용자의 수동 테스트(manual test)를 통해서 이루어지거나 Capture 및 Replay 방법을 이용해 테스트를 수행하고 있다.

이에 본 논문에서는 웹 어플리케이션의 기능 테스트를 자동화 하는 방법을 제안한다. 이를 위해 소스코드와 HTML 코드 분석을 통해 웹 어플리케이션 분석 모델을 생성하는 방법, 분석 모델로부터 테스트 자동화에 필요한 정보를 추출하는 방법, 추출된 정보로부터 테스트 준비과정을 자동화하는 방법, 그리고 준비과정에서 자동으로 생성한 테스트 드라이버와 테스트 데이터를 이용해 테스트 실행 과정을 자동화하는 방법을 제안한다. 또한 이러한 방법을 기반으로 구현한 웹 어플리케이션 테스트 자동화 시스템의 프로토타입을 소개한다.

본 논문은 다음과 같이 구성된다. 2장에서는 웹 어플리케이션 테스트와 기존의 연구들에 대해 살펴본다. 3장에서는 웹 어플리케이션 테스트 자동화에 관한 내용을 기술한다. 4장에서는 프로토타입 시스템과 사례 연구를 기술하고, 5장에서 결론 및 향후 연구방향을 기술한다.

2. 관련연구

2.1 웹 어플리케이션 테스트

현재 웹 어플리케이션은 (그림 1)에서 보여지는 것과 같이 다양한 컴포넌트로 구성되며, 대부분은 다계층 아키텍처를 이용해 구성된다. 클라이언트 측은 웹 브라우저, 스크립트 컴포넌트, 플러그 인 컴포넌트 등을 포함한다. 웹 서버 측은 HTML, PHP, ASP, CGI등으로 구현된 어플리케이션 로직을 포함한다. 그리고 어플리케이션 서버 측에는 응용 서버 컴포넌트를 포함하고, EIS 서버는 데이터관리를 위한 데이터 서비스 컴포넌트 및 기존의 레거시(legacy) 컴포넌트를 포함한다[2, 3].

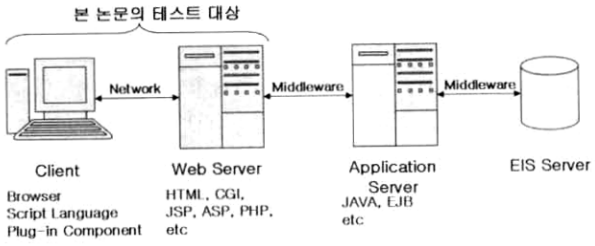
웹 어플리케이션을 테스트 하려면 구성하는 각 컴포넌트에 대한 단위 테스트를 수행한 후 통합 테스트를 수행해야 한다. 그러나 웹 어플리케이션은 (그림 1)과 같이 다양한 언어와 기술이 통합적으로 사용되며, 정확한 단위로 분할하기 어렵기 때문에 전통적인 테스트 기법을 적용하기 어렵다. 또한 웹 어플리케이션은 다양한 플랫폼 환경에 분산될 수 있으며, 인터넷을 통해 약결합(loosely-coupled)되어 있어 테스트를 수행하기 어렵고, 실행 흐름에 따라 다양한 웹 페이지를 생성하기 때문에 테스트 케이스 추출이 쉽지 않다[2, 5].

본 논문에서는 (그림 1)의 웹 어플리케이션의 구성 컴포넌트 중에서 클라이언트 측 페이지(client-side page)의 사용자 인터페이스 테스트 및 기능 테스트와 서버 측 페이지(server-side page)의 어플리케이션 로직에 대한 기능 테스트를 자동화하기 위한 방법에 대하여 기술한다.

2.2 웹 어플리케이션 테스트에 관한 연구

기존의 웹 어플리케이션 기능 테스트와 관련된 노력은 대부분 다른 테스트 방법을 웹 어플리케이션에 적용하거나, 웹 어플리케이션의 테스트 모델을 만드는 것들이 대부분이다.

논문 [4]는 기존의 객체지향 테스트 모델을 웹 어플리케이션에 적용한 새로운 테스트 모델을 소개한다. 웹 어플리케이션의 테스트를 위해 구성 페이지를 객체로 보고 객체관점(object perspective), 행위관점(behavior perspective), 구조관점(structure perspective)으로 분석하여 테스트 모델을 생성한다. 객체관점에서는 웹 어플리케이션을 객체 관계 다이어그램(ORD: object-relation diagram)으로 표현한다. 이 ORD는 테스터에게 어플리케이션의 구조를 이해하는데 도움을 주고, 특정페이지 변경 시 파급효과(ripple effect)를 분석하는데 도움을 준다. 또한 웹 어플리케이션의 테스트 방법이나 도구에 적용할 수 있다고 말하고 있다. 행위관점에서는 웹 어플리케이션의 페이지 네비게이션 다이어그램(PND: page navigation diagram), 객체 상태 다이어그램(OSD: object state diagram)으로 표현한다. PND를 이용하여 테스트 케이스 및 테스트 시나리오를 추출할 수 있고, 접근성(reachability)이나 데드락(deadlock)을 분석하는데 도움을 준다. 또한 OSD를 이용하여 각 페이지의 상태에 따른 테스트 케이스를 생성하는 방법을 설명한다. 구조 관점에서의 웹 어플리케이션은 제어 흐름(control flow), 데이터 흐름(data flow)을 분석하여, 데이터 흐름에 따른 테스트 케이스 생성 방법에 대해



(그림 1) 웹 어플리케이션 구성

설명한다. 이 논문의 테스트 모델은 웹 어플리케이션을 기존의 객체지향 어플리케이션의 테스트 방법을 적용하여 다양한 관점에서 테스트 방법을 소개하지만 분석 과정에서 많은 오버헤드를 갖는다는 단점이 있다. 웹 어플리케이션이 복잡해질수록 물리적인 단위로 나누고 분석하는데 많은 시간과 노력이 필요하기 때문이다. 또한 이렇게 물리적인 단위로 나누더라도 어플리케이션을 변경할 경우 분석과 테스트 케이스 생성 과정을 반복 수행해야 하는 번거로움이 있으며, 분석하는 사람의 역량에 의해 테스트에 대한 신뢰성의 기복이 심하다는 단점이 있다.

[5]는 기존의 구조적 테스트 기법을 웹 어플리케이션에 적용한 테스트 모델을 소개한다. 웹 어플리케이션의 페이지를 객체로 보고 각 페이지의 HTML과 XML의 데이터 흐름과 제어 흐름을 분석하여 테스트 케이스를 생성하고, 이를 이용한 화이트 박스 테스트(white-box test) 기법을 소개하고 있다. 이 논문 또한 ORD형태로 웹 어플리케이션을 분석한 후 데이터 흐름에 따른 다양한 테스트 케이스 생성 방법을 소개하지만, 이것 역시 웹 어플리케이션의 분석 과정에서 많은 오버헤드를 갖는다. 웹 어플리케이션 자체가 다양한 기술로 구현될 수 있으며, 이러한 복잡도가 증가할수록 데이터의 흐름을 분석하기가 쉽지 않기 때문이다.

[6]은 웹 서버 측의 사용자 세션 데이터(user-session data)을 분석하여 테스트 케이스를 생성하고 자동으로 테스트를 수행하는 도구를 구현한 사례를 보여준다. 사용자 세션 데이터는 서버에 접근한 접근 기록(access log)를 의미하며, 이는 실제 사용자의 사용 패턴을 나타낸다. 논문 [6]에서는 이러한 정보를 이용하여 페이지들간의 관계를 파악하고, 페이지 요청에 사용되는 파라메터 값 쌍을 분석한다. 이러한 정보를 이용하여 테스트 케이스 및 테스트 시나리오를 생성하여 테스트를 자동화한다. 그리고 테스트 결과 판단을 위해 온전한 사용사례에 대한 결과를 저장하여 테스트 수행 시 산출되는 결과와 비교하여 테스트 결과를 산출한다. 이 논문은 웹 어플리케이션의 테스트 자동화를 구현한 몇 안 되는 연구이다. 하지만 이 논문의 방법은 서버에서 사용자의 접근에 대한 로그를 기록하지 않는 경우에 웹 서버의 기능을 확장해야 하는 단점이 있다. 또한 단순히 서버 측의 로그의 조합을 기반으로 테스트를 수행하기 때문에 사용자가 사용하지 않은 부분에 대해 테스트의 수행을 장담할 수 없는 단점이 있다.

[7]은 웹 어플리케이션의 테스트를 위해 객체 지향 테스트 모델을 소개하고 단위 테스트 및 통합 테스트를 수행하는 방법을 소개한다. 웹 어플리케이션의 테스트 모델은 UML 클래스 다이어그램으로 표현한다. 단위 테스트를 위해 각 페이지에 대한 결정 테이블(decision table)을 작성하고 각 입력에 대한 결과 값을 채우는 방법으로 테스트를 수행한다. 통합 테스트를 위해서는 유스케이스 다이어그램으로부터 일련의 테스트 시나리오를 추출하여 테스트 케이스로 이용하는 방법을 이용한다. 그러나 이 방법은 많은 경우에 테스트 과정에서 사용자의 개입을 요구한다. 테스트 수행을 위해 모든 대상 페이지에 대해 결정 테이블을 사용자가 직접 작성하는 부담을 갖고 있을 뿐만 아니라 대상 페이지의 변경 시 작성해 놓은 결정 테이블을 재활용할 수 없다는 단점도 갖고 있다.

그리고 [8]에서는 웹 어플리케이션의 HTML코드의 분석과 자바 스크립트를 이용한 사용자 인터페이스의 테스트 자동화 방법을 소개하고 있다. 이 논문은 브라우저 객체를 이용하여 HTML 웹 페이지와 사용자 입력 부분에 대한 테스트 케이스를 자동으로 실행할 수 있는 방법을 제시하고, 자바스크립트로 매핑하여 자동화하는 방법을 소개한다. 이 방법은 웹 어플리케이션의 사용자 인터페이스에 대한 테스트 수행을 자동화 할 수 있다. 그러나 이 방법 역시 많은 사용자의 개입을 요구한다. 이 논문은 사용자 인터페이스의 테스트 자동화에 대한 장점을 가지지만, 테스트 케이스에 해당하는 스크립트를 사용자가 직접 작성해야 한다. 또한 사용자 인터페이스를 테스트하는 방법을 제시할 뿐 웹 어플리케이션의 기능 테스트에 대한 방법을 제시하지는 못한다.

위와 같은 방법들은 테스트 자동화에 적용하기 어렵다. 우선 앞서 설명한 바와 같이 웹 어플리케이션은 다양한 서버에 분산되어 있고, 약결합되어 있어 컴포넌트의 분할이 쉽지 않다. 또한 웹 어플리케이션은 실행 흐름과 그 기능에 따라 동적으로 생성되는 경우가 많기 때문에 각각의 상황을 고려하여 테스트가 수행되어야 한다. 하지만 웹 어플리케이션의 특징을 고려하지 않은 채 기존의 테스트 방법을 그대로 적용하는 것에서 정확한 테스트를 기대할 수 없다. 또한 테스트 자동화에 관련된 대부분의 연구들은 여전히 사용자의 개입을 요구하고 있거나 충분한 테스트를 수행하지 못한다는 단점이 있다.

본 논문에서는 위와 같은 문제를 해결하기 위하여 테스트 대상 페이지 및 관련 페이지의 HTML 코드와 소스코드를 분석하여 테스트 자동화에 필요한 모든 정보와 테스트 케이스를 추출하는 방법을 연구하였다. 이 정보를 기반으로 테스트 드라이버와 테스트 데이터를 자동으로 생성한다. 이는 테스트 대상을 사용자 인터페이스를 가진 클라이언트 측 페이지와 웹 어플리케이션의 기능을 수행하는 서버 측 페이지로 나누어 테스트 드라이버를 생성하며, 생성된 테스트 데이터를 이용해 테스트 수행까지 자동화한다. 또한 테스트 수행을 위해 관련된 모든 파일의 배치 또한 자동화하여 테스트 환경 구축을 자동화한다. 이는 기존의 연구들에서 갖는 단점인 사용자의 개입을 최소화 할 수 있다.

3. 웹 어플리케이션 테스트 자동화

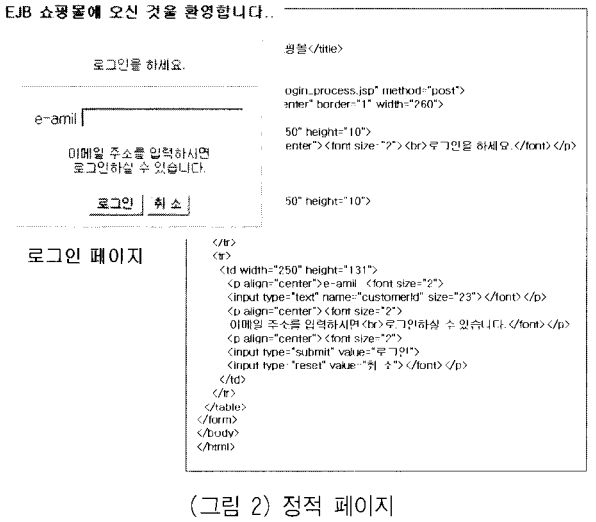
3.1 테스트 대상

본 논문에서는 웹 어플리케이션의 구성 컴포넌트 중에서 클라이언트의 브라우저에 나타나는 클라이언트 측 페이지와 웹 서버에서 동작하는 어플리케이션 로직을 담고 있는 서버 측 페이지를 테스트 대상으로 설정하고 테스트를 자동으로 수행하는 방법에 대해 연구하였다.

3.1.1 클라이언트 측 페이지 테스트 전략

클라이언트 측 페이지는 클라이언트의 브라우저 상에 나타나는 페이지를 말한다. 이 페이지는 크게 정적 페이지(static page)와 파라메터에 의해 내용이 결정되는 동적 페이지(dynamic page)로 구분된다. 다시 말해 정적 페이지는 다음 (그림 2)의 로그인 페이지와 같이 순수 HTML로만 구현된 것이며, 동적 페이지는 (그림 3)의 인터넷 쇼핑몰 어플리케이션의 상품 보기 페이지와 같이 전달되는 파라메터에 따라 내용이 변경되는 페이지를 말한다.

클라이언트 측 페이지의 테스트는 두 가지 관점에서 이루어져야 한다. 첫 번째는 요청된 클라이언트 페이지가 정확히 브라우저 상에 나타나는지 혹은 예외적인 상황에서 바르게 대처하는지 확인하는 사용자 인터페이스 테스트이다. 이는 브라우저 상에 요청된 페이지가 단순히 로드 되는 지 확인하는 것을 의미하는 것이 아니라 구현 시 의도했던 위치에 의도한 컨텐츠가 정확하게 로드 되는지 확인해야 하는 것을 의미한다. 이를 위해 (그림 4)와 같은 방법으로 테스트를 수행한다. 두 번째 테스트 관점은 클라이언트 페이지가 갖는 기능에 대한 테스트이다. 클라이언트 측 페이지의 기능은 자바스크립트와 같은 스크립트 언어, ActiveX와 같은 플러그 인 컴포넌트의 기능 등을 의미한다. 예를 들어 로그인 페이지에서 스크립트 언어로 작성된 ID와 패스워드 입력 확인 기능과 같은 것이다. 클라이언트 페이지의 기능은 웹 어플리케이션에서 매우 중요한 부분을 차지하지만 많은 경우 그에 관한 테스트가 제대로 이루어지지 않는다. 또한 수시로 변경되는 클라이언트 페이지의 테스트에 있어 자동화된 방법은 필수적이다. 본 논문에서는 이러한 클라이언트 측 페이지의 기능 중에서 스크립트에 의한 입력 제어 기능만을 대상으로 테스트 자동화 방법을 제안하며, (그림 5)와 같은 방법으로 테스트를 수행한다.

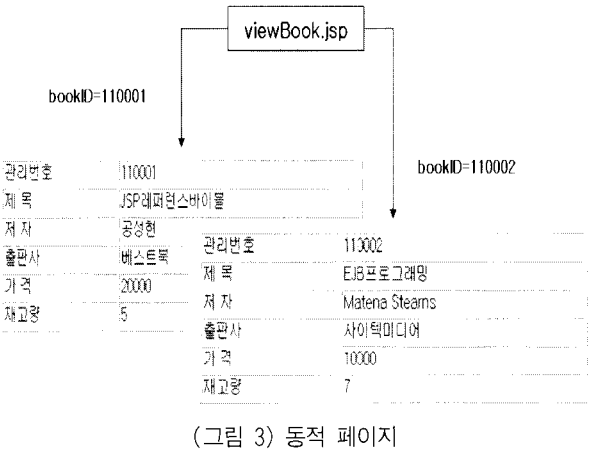


(그림 2) 정적 페이지

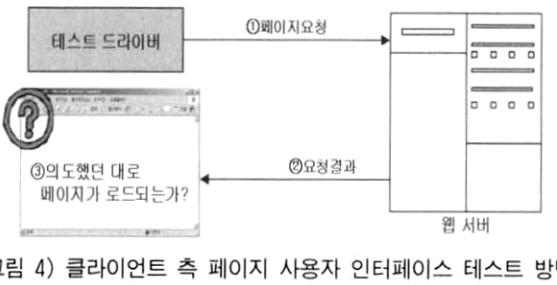
3.1.2 서버 측 페이지 테스트 전략

서버 측 페이지는 클라이언트의 브라우저 상에 나타나지는 않지만 서버 내에서 어플리케이션 로직을 수행하는 페이지를 의미한다. 예를 들어 인터넷 쇼핑몰 어플리케이션의 상품 등록과 관련된 기능을 보면, (그림 6)과 같이 상품 정보 등록 페이지→상품 정보 등록 처리 페이지→등록 확인페이지의 순서로 진행이 된다. 이때 상품 정보 등록 처리 페이지가 서버 측 페이지에 해당한다.

서버 측 페이지의 경우 해당 로직을 구현한 페이지를 서버에 요청했을 때 구현된 기능이 정확히 수행되는지 확인하는 방법으로 테스트를 수행한다. 예를 들어 앞서 설명한 상품 등록 페이지의 경우 상품 정보를 파라메터로 주고 페이지를 요청했을 때 등록 처리 페이지의 기능이 정확히 수행된 다음, 등록 확인페이지로 넘어가서 결과를 확인할 수 있어야 하며, 만일 등록 처리 페이지가 특정 사람들에게 메일을 발송할 경우 메일이 도착했는지 확인해야 한다.

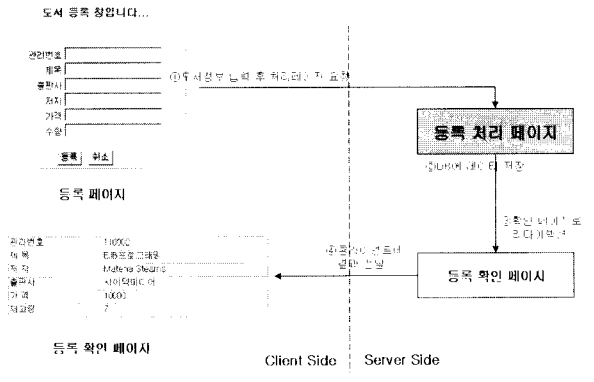


(그림 3) 동적 페이지



(그림 4) 클라이언트 측 페이지 사용자 인터페이스 테스트 방법

(그림 5) 클라이언트 측 페이지 입력 제어 기능 테스트 방법



(그림 6) 서버 측 페이지

본 논문은 이와 같은 방법으로 클라이언트 측 페이지와 서버 측 페이지를 테스트하는 자동화된 방법을 제안한다. 일반적으로 테스트 자동화는 4 단계로 나누어 수행된다. 첫 번째 단계에서는 어플리케이션을 분석하여 분석 모델을 생성하고, 이를 기반으로 테스트 대상과 테스트 자동화에 필요한 정보를 추출한다. 두 번째 단계는 테스트 준비 단계로 첫 번째 단계의 산출물을 기반으로 테스트 드라이버와 테스트 데이터를 자동으로 생성하고 배치의 자동화를 통해 테스트 수행 환경을 자동으로 구축한다. 세 번째 단계는 테스트 수행 단계로 자동으로 생성된 테스트 드라이버와 테스트 데이터를 이용해 테스트를 수행한다. 마지막 단계는 테스트 결과 판단 단계로 수행된 테스트 결과를 판단한다. 본 논문에서는 웹 어플리케이션의 테스트 자동화를 위해 분석 단계와 준비 단계, 그리고 테스트 실행 단계를 자동화한다. 다음 절부터는 각 단계에 대한 자세한 설명을 기술한다.

3.2 웹 어플리케이션 분석 단계(Web Application Analyzing Stage)

웹 어플리케이션 분석 단계에서는 테스트 대상인 웹 어플리케이션을 분석하여 구체적인 테스트 대상과 테스트 케이스를 추출하는 작업을 한다. 이 단계에서는 4가지 작업이 수행된다.

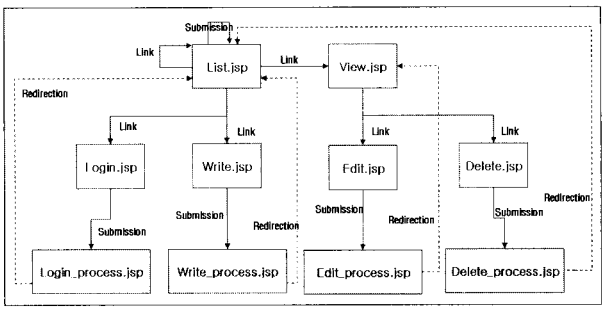
- 웹 어플리케이션을 구성하는 페이지들 간의 관계를 파악하여 분석 모델 생성
- 분석 모델을 기반으로 테스트 대상 파악
- 분석 모델을 기반으로 테스트 케이스 추출
- 테스트 케이스 별 파라메터 정보 및 제약(constraint) 정보 추출

3.2.1 웹 어플리케이션 분석 모델 생성

본 논문에서는 테스트를 위해 웹 어플리케이션의 분석 모델을 자동으로 생성한다. 웹 어플리케이션의 분석 모델은 어플리케이션을 구성하는 페이지와 그들 간의 요청 관계를 나타낸다.

[웹 어플리케이션 분석 모델] 웹 어플리케이션 분석 모델은 노드(node)와 에지(edge)로 구성된다.

- 노드
  - 의미: 웹 어플리케이션을 구성하는 개별 페이지
  - 종류: 클라이언트 측 페이지, 서버 측 페이지
  - 표현: 흰색 사각형(클라이언트 측 페이지), 회색 사각형(서버 측 페이지)
- 에지
  - 의미: 페이지들 간의 요청 관계 (화살표 방향으로 페이지 요청)
  - 종류: 링크(link), 서브미션(submission), 리다이렉션(redirection)
  - 표현: 실선화살표(링크, 서브미션), 점선화살표(리다이렉션)



(그림 7) 게시판 어플리케이션의 분석 모델

예를 들어 게시판 어플리케이션의 분석 모델은 그림 7과 같이 표현될 수 있다. (그림 7)에서 6개의 클라이언트 측 페이지와 4개의 서버 측 페이지를 볼 수 있다. 그리고 실선화살표로 나타낸 에지가 사용자의 액션에 의한 페이지 요청, 그리고 점선화살표가 사용자의 액션 없이 자동으로 페이지가 이동하는 것을 나타낸다.

[웹 어플리케이션 분석 모델 생성 방법]

1. 웹 어플리케이션의 페이지 목록으로부터 노드 추출
2. 노드의 종류 결정
   - 2.1 페이지가 HTML 코드를 갖고 있으면 클라이언트 측 페이지
   - 2.2 페이지가 HTML 코드를 갖고 있지 않으면 서버 측 페이지
3. 에지의 종류 결정
   - 3.1 HTML 코드를 분석(파싱: parsing)하여 링크 및 서브미션 정보 추출
     - 3.1.1 HTML 코드에서 <a> 태그로부터 링크 정보 추출
     - 3.1.2 HTML 코드에서 <form> 태그로부터 서브미션 정보 추출
   - 3.2 소스 코드를 파싱하여 리다이렉션 정보 추출

예를 들어 게시판 어플리케이션의 로그인 페이지(Login.jsp)의 HTML 코드를 보면 다음과 같은 내용을 포함한다.

```
<form method="post" action="login_process.jsp"…>
  <input type="text" name="USERID" maxlength="12"…>
  <input type="password" name="USERPW"…>
  …
</form>
```

여기서 <form> 정보를 분석하면 로그인 처리 페이지(Login_process.jsp)와의 관계를 추출할 수 있고, 로그인 처리 페이지를 요청할 때 전달되어야 할 파라메터 정보를 추출할 수 있다. 링크 정보도 마찬가지 방법으로 추출한다. 예를 들어 게시물 목록 페이지의 HTML코드는 제목와 같은 정보를 포함한다. 이를 분석하면 게시물 목록 페이지가 게시물 조회 페이지와 관계가 있음을 알 수 있고, 게시물 조회 페이지를 요청할 때 전달되어야 하는 파라메터에 대한 정보를 추출할 수 있다. 그러나 리다이렉션 정보는 해당 페이지를 구현하고 있는 소스 코드로부터 추출한다. 이는 리다이렉션이 사용자의 액션에 의한 페이지 이동이 아니며, 서버 측 페이지에서 기능 수행 후 다른 페이지로 자동 이동할 때 이용하기 때문에 HTML 코드를 포함하지 않는다. 예를 들어 로그인 처리 페이지는 서버 측 페이지이며, 어떠한 HTML 코드도 포함하고 있지 않기 때문에 HTML 코드 분석 방법을 이용할 수 없다. 그러나 소스 코드를 분석해보면 리다이렉션 정보response.sendRedirect("list.jsp")에서 게시물 목록 페이지(List.jsp)와 요청 관계가 있음을 알 수 있다. 본 논문에서는 이와 같이 웹 어플리케이션 HTML 코드와 소스 코드를 분석하여 웹 어플리케이션의 분석 모델을 생성한다. 다음 (그림 8)은 게시물 목록, 로그인, 로그인 처리 페이지의 관계를 추출하는 과정을 보여주고 있으며, 이러한 방법을 통해 그림 7에서 "List.jsp → Login.jsp → Login_process.jsp ⇒ List.jsp"와 같은 페이지들 간의 관계를 파악할 수 있다.

이미 많은 연구들이 웹 어플리케이션 테스트를 위해 여러 분석 모델을 제안하고 있다. [4], [5] 연구의 경우 ORD를 이용한 분석 모델을 제안하고 있지만 자동화된 방법을 말하고 있지 않다. [7]의 경우 UML의 클래스 다이어그램 형태의 분석 모델을 이용하지만 설계 단계의 산출물을 이용하거나 사용자의 분석 작업을 요구한다. 그리고 [6]의 경우 사용자 세션 정보를 기반으로 웹 어플리케이션의 구조를 분석한다. 하지만 이 방법의 경우 사용자가 접근하지 않는 페이지에 대해서 분석하지 못한다는 단점이 있다. 본 논문의 경우 소스코드와 HTML을 분석하여 웹 어플리케이션의 분석 모델을 생성하므로 모든 페이지에 대해 자동화된 방법으로 분석이 가능하다는 장점을 가진다.

3.2.2 테스트 대상 선정

본 논문의 테스트 대상은 3.1절에서 설명한 바와 같이 웹 어플리케이션을 구성하는 각각의 페이지 즉, 클라이언트 측 페이지와 서버 측 페이지이다. 이는 웹 어플리케이션 분석 모델에서 각 노드에 대한 정보를 이용하여 알 수 있다.

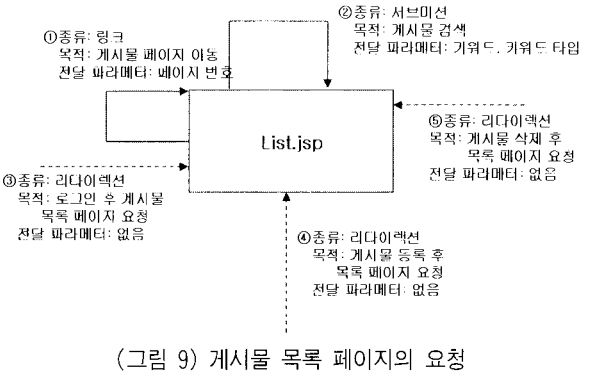
클라이언트 측 페이지의 사용자 인터페이스 테스트는 해당 페이지에 대한 각각의 요청을 테스트 대상으로 삼는다. 예를 들어 (그림 9)의 게시물 목록 페이지(List.jsp)를 보면 해당 페이지 요청에 대한 에지가 링크, 서브미션, 그리고 리다이렉션이 모두 존재한다. 각각의 에지에 대해 자세히 살펴보면 다음과 같다.

위의 그림에서 볼 수 있듯이 게시물 목록 페이지의 요청은 ①과 같은 게시물 목록의 페이지 이동과 관련된 링크, ②와 같은 게시물 검색을 위한 서브미션, 그리고 ③과 같은 로그인 처리 후, ④와 같은 게시물 등록 후, ⑤와 같은 게시물 삭제 후의 리다이렉션으로 나눌 수 있다. 이는 게시물 목록 페이지가 갖는 개별적 기능으로 테스트 수행 시 모두 테스트되어야만 한다. 그러나 ①의 경우 페이지 번호를 파라메터로 전달해 해당 페이지의 게시물 목록을 요청하는 기능을 수행하고, ②의 같은 경우 키워드와 키워드 타입을 파라메터로 전달하여 검색 기능을 수행한다. 그리고 ③~⑤의 경우 모두 다 전달하는 파라메터 없이 게시물 목록 페이지를 요청한다. 따라서 게시물 목록 페이지에 대한 기능은 전달되는 파라메터의 종류에 따라 구분할 수 있으며 ①, ②, ③의 요청을 테스트하는 것으로 모든 기능을 테스트할 수 있다. 이와 같은 방법으로 클라이언트 측 페이지가 갖는 사용자 인터페이스를 테스트하기 위해서는 웹 어플리케이션의 분석 모델에서 해당 페이지의 요청 즉, 에지 정보를 이용해 테스트 대상을 선정한다. 또한 에지의 파라메터 정보를 통해 동일 기능을 수행하는지 여부를 판단하며, 동일 기능일 경우 테스트 대상에서 제외하는 방식으로 중복을 제거한다.

클라이언트 측 페이지의 기능 테스트를 위한 테스트 대상 선정은 간단하다. 본 논문에서는 클라이언트 측 페이지의 기능을 사용자의 입력 제어 기능으로 한정하였기 때문에 사용자 입력 폼을 갖는 페이지에서 입력 폼의 입력 제어 기능이 테스트 대상이다.

마지막으로 서버 측 페이지의 테스트는 해당 페이지가 갖는 개별 로직이 대상이다. 해당 페이지가 갖는 기능은 전달되는 파라메터의 종류에 따라 달라질 수 있기 때문에 서버 측 페이지의 요청을 나타내는 에지 정보의 파라메터 종류에 따라 테스트 대상을 달리한다. 이는 클라이언트 측 페이지

(그림 8) 정보 추출 과정



(그림 9) 게시물 목록 페이지의 요청

의 사용자 인터페이스 테스트와 유사한 개념이다. 예를 들어 (그림 7)에서 "Login_process.jsp"에 대한 테스트 대상은 서브미션 에지를 통해 수행되는 "Login.jsp → Login_process.jsp"의 페이지 요청을 통해 파악될 수 있다.

3.2.3 테스트 케이스 생성

테스트 케이스는 웹 어플리케이션 분석 모델에서 에지의 파라메터 정보를 이용해 생성한다. 예를 들어 (그림 7)의 게시물 목록, 로그인, 게시물 처리 페이지들의 관계에서 에지로 표현되는 링크/서브미션/리다이렉션 정보를 보면 각각의 테스트 대상에 대한 파라메터의 정보를 추출할 수 있다. 그러나 에지로부터 추출되는 정보는 단순히 전달되는 파라메터의 이름이다. 기본적으로 HTTP 프로토콜에서는 모든 정보를 문자열 형태로 전달하며, 파라메터를 전달받는 페이지에서 정확한 타입으로 캐스팅(casting)해서 사용한다. 따라서 HTML 코드를 분석하더라도 파라메터의 타입은 알 수 없다. 테스트 데이터를 생성할 때 파라메터의 타입을 모른다면 정밀한 데이터를 생성할 수 없으며, 이로 인해 테스트의 효율이 떨어진다. 그러므로 전달되는 파라메터의 타입을 알아야 한다. 본 논문에서는 각 파라메터의 타입을 추출하기 위해 소스 코드를 분석한다. JSP 기반의 웹 어플리케이션의 경우 넘겨지는 파라메터를 캐스팅하여 사용하기 때문에 캐스팅 정보를 이용해 정확한 타입 정보를 추출할 수 있다. 예를 들어 로그인 처리 페이지의 경우 전달되는 파라메터를 다음과 같은 형태로 전달받기 때문에 USERPW라는 파라메터의 타입이 정수형임을 알 수 있다.

```
int USERPW = Integer.parseInt(request.getParameter("USERPW"));
```

이렇게 추출된 정보는 테스트 케이스를 생성하기 위한 기본 정보이다. 이 정보를 기반으로 테스트 준비 단계의 테스트 데이터 생성에서 만들어지는 데이터를 이용하여 테스트 케이스를 완성한다.

3.2.4 파라메터 제약 정보 추출

마지막으로 파라메터 제약 정보는 HTML 코드를 분석하여 추출한다. 제약정보를 추출하는 이유는 모든 파라메터에 대해 제약 정보를 추출할 수 없지만, HTML 코드 상에 파라메터의 제약 정보가 나타나는 경우 이 제약 정보를 추출함으로써 테스트 데이터를 생성할 때 좀 더 정밀한 데이터를 생성할 수 있기 때문이다. 일반적으로 웹 어플리케이션을 개발 할 때 HTML 코드에 입력 값에 대한 제약 정보를 삽입할 수 있다. 예를 들어 <input type="text" name="ID" maxlength="8">이라는 코드를 보면 입력되는 데이터의 길이가 8자리를 넘지 않아야 한다는 것을 알 수 있다. 또한 셀렉트 박스(Select Box), 체크 박스(Check Box), 라디오 버튼(Radio Button)의 경우 입력에 대한 값의 범위가 한정적이기 때문에 테스트 데이터 생성 시 제한된 값의 입력을 통해 정밀한 테스트 데이터를 생성할 수 있다. 그러나 위와 같은 제약 사항은 테스트 케이스 중에서 서브미션에 해당하는 정보만을 추출할 수 있으며, 링크나 리다이렉션 정보에서는 추출할 수 없다.

3.3 테스트 준비 단계(Test Preparing Stage)

테스트 준비 단계에서는 웹 어플리케이션 분석 단계에서 추출된 정보를 기반으로 크게 3가지 작업이 수행된다.

- 테스트 데이터 자동 생성
- 테스트 드라이버 자동 생성
- 테스트 드라이버 및 테스트 데이터 자동 배치

3.3.1 테스트 데이터 자동 생성

테스트 데이터는 테스트 대상에 대한 테스트 케이스의 기본 정보를 이용해 생성한다. 여기서 필요한 것은 페이지 요청 시 전달되는 파라메터의 이름, 타입, 그리고 파라메터의 제약 정보이다. (그림 10)은 예제 코드로부터 추출한 테스트 케이스의 기본 정보를 보여준다. 예제 코드의 HTML로부터 파라메터 종류와 제약 정보를 추출할 수 있으며, 소스코드로부터 문자열과 정수형의 타입을 가짐을 알 수 있다.

이러한 정보를 기반으로 동치 영역 분할(Equivalence Partitioning) 기법을 통해 테스트 데이터를 생성한다. 예를 들어 (그림 10)의 소스코드에 대한 테스트 데이터의 생성 방법은 다음과 같다.

- kinds 파라메터: kinds 셀렉트 박스로 전달되는 파라메터의 경우 전달되는 값의 범위가 1,2,3으로 명확하기 때문에 전달될 수 있는 값과 전달될 수 없는 값으로 동치 영역을 분할해 테스트 데이터를 생성한다. 전달될 수 있는 값의 경우 셀렉트 박스 안에 포함되는

```
<Select name="kinds">
  <option value=1 selected>도서
  <option value=2>음반
  <option value=3>티켓
</Select>
<input name="count" maxlength="4" size="8">
<input name="name" size="8">
```

```
Int kinds = Integer.parseInt(request.getParameter("kinds"));
Int count = Integer.parseInt(request.getParameter("count"));
String name = request.getParameter("name");
```

변수명	타입	제약사항
Kinds	Int	1,2,3
Count	Int	Length(4)
Name	String	

(그림 10) 테스트 케이스 예제

값이며, 전달될 수 없는 값은 NULL 값이나 제약 사항 안에 포함되지 않는 임의의 값이다. 전달될 수 없는 값의 경우 일반적인 사용에서는 나타나지 않지만 사용자가 임의로 브라우저 상에 직접 URL로 입력하여 해당 페이지에 접근할 수 있기 때문에 이러한 예외적인 상황에 대한 테스트가 수행되어야 한다.

- count 파라메터: count 파라메터의 경우 정수형이며 최대 가질 수 있는 값의 범위가 네 자리 수이기 때문에 정상 범위인 -9999~+9999와 그 이외의 범위로 나누어 데이터를 생성한다. 이때 데이터의 영역을 -9999미만, -9999이상, 0, +9999이하, +9999이상의 다섯 가지 영역으로 나누어 데이터를 생성한다. 또한 예외적인 상황의 테스트를 위해 정수형 데이터와 정수형 데이터가 아닌 값으로 나누어 테스트 데이터를 생성한다.
- name 파라메터: 문자열 데이터의 경우 여러 상황으로 나누어 볼 수 있다. 단순 문자열, URL, 데이터 스트림(Data Stream) 등등 문자열을 통해 전달될 수 있는 여러 가지 경우가 있기 때문에 전달될 수 있는 데이터의 패턴을 분석하여 테스트 데이터를 생성한다.

다음 표는 위의 테스트 케이스로부터 생성한 데이터의 일부분을 나타낸다.

본 논문에서는 현재 HTML 코드와 소스코드 정보를 기반으로 추출한 테스트 케이스 정보를 이용해 테스트 데이터를 생성한다. 정밀한 테스트 데이터를 생성하고, 불필요한 데이터를 배제하기 위해 파라메터 정보, 타입 정보, 그리고 제약사항을 이용하였다. 본 논문의 연구에서는 이러한 테스트 데이터를 자동으로 생성하기 때문에 사용자의 개입을 최소화할 수 있다. 그러나 보다 더 정밀한 데이터를 생성하기 위해서는 어플리케이션의 스펙이나, 데이터 흐름 분석이 필요하다. 웹 어플리케이션의 각 페이지에 대한 스펙이 주어질 경우 추가적인 제약 정보를 이용해 좀 더 정확한 데이터를 생성할 수 있다. 또한 웹 어플리케이션의 데이터 흐름 분석을 이용하면, 각 파라메터 별 데이터 범위 및 수행 흐름을 알 수 있어 정확한 오류의 범위를 찾아낼 수 있을 것이다. 그러나 스펙의 경우 테스트 수행 이전에 사용자가 스펙을 기술해야 하며, 데이터 흐름 분석의 경우 추가적인 분석 도구를 필요로 한다. 향후 본 연구에서는 웹 어플리케이션의 스펙을 기술하는 방법, 이러한 스펙을 이용하여 테스트 데이터를 생성하는 방법 및 데이터 흐름 분석을 통해 테스트 데이터를 생성하는 방법에 대한 연구를 수행할 예정이다.

〈표 1〉 테스트 데이터

No	Kinds	Count	Name
1	1	100	"String"
2	'A'	80	"../test.jsp"
3	2	100000	"123456789"
4	2	55	null
5	7	100000	null
...	...	...	...

3.3.2 테스트 드라이버 생성

테스트 드라이버는 다음과 같이 세 가지 형태로 생성된다.

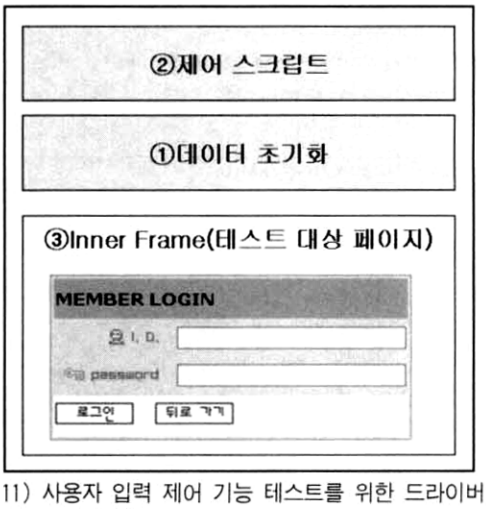
- 클라이언트 측 페이지의 사용자 인터페이스 테스트 수행을 위한 테스트 드라이버
- 서버 측 페이지의 기능 테스트 수행을 위한 테스트 드라이버
- 클라이언트 측 페이지의 기능 테스트 수행을 위한 테스트 드라이버

서버 측 페이지의 기능 테스트와 클라이언트 측 페이지의 사용자 인터페이스 테스트에 적용될 테스트 드라이버는 웹 서버에 해당 페이지를 요청하고, 결과로 전달되는 HTML 스트림을 저장한다. 이 유형의 테스트 드라이버는 다음과 같이 세 부분으로 구성된다.

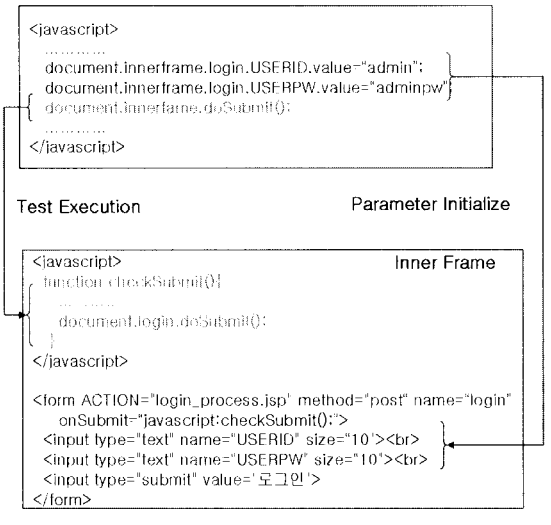
- 파라메터 초기화 부분: 생성된 테스트 데이터를 읽어들여 파라메터의 값을 초기화
- 구동 부분: 초기화된 파라메터를 갖고 URL을 조합한 다음, 페이지 요청을 수행
- 결과 저장 부분: 페이지 요청에 대한 결과를 저장

페이지 요청은 테스트 대상 페이지 + 파라메터 값 쌍(Parameter-Value Pair)으로 이루어진다. 예를 들어 게시판 어플리케이션의 게시물 등록 처리 페이지의 테스트를 위해 "write_process.jsp?subject=제목&content=내용…"과 같은 URL로 서버에 페이지 요청을 하면, 서버에 존재하는 "write_process.jsp"가 구동하고 그 수행 결과가 반환된다. 계속해서 그 수행 결과를 분석하여 기능이 정확하게 동작하는지 여부를 판단한다. 이때 테스트 데이터의 수만큼 반복해서 테스트를 수행하게 된다. 테스트의 수행 과정은 다음 절의 테스트 실행 단계에서 자세히 다룬다.

클라이언트 측 페이지의 사용자 입력 제어 기능을 테스트하기 위한 테스트 드라이버의 형태는 (그림 11)과 같다.



(그림 11) 사용자 입력 제어 기능 테스트를 위한 드라이버 구조



(그림 12) 스크립트를 이용한 이벤트 핸들링

이 드라이버는 3개의 부분(파라메터 초기화, 제어 스크립트, 테스트 대상 파일)으로 구분된다. ①부분은 파라메터를 초기화 하는 부분이다. 테스트 데이터 자동 생성 모듈에서 생성된 데이터 파일을 읽고, 정보 추출 모듈로부터 추출된 파라메터에 기본 값을 설정하는 부분이다. ②부분은 스크립트로 작성된 부분으로 테스트 대상 페이지의 입력을 제어하고, 테스트를 수행하는 부분이다. Jscript, Java Script, VB Script 등과 같은 스크립트 언어는 다소 제한적이지만, (그림 12)와 같이 요청 기반 이벤트 핸들링을 브라우저에서 시뮬레이션하는데 사용될 수 있으며, 논문 [8]에서는 이러한 방법을 이용하여 웹 어플리케이션의 인터페이스 테스트 자동화 시스템을 구축한 사례를 제시하고 있다. 하지만 논문 [8]의 경우 테스트를 위해 일련의 스크립트를 사용자가 작성해야 하며, 작성된 스크립트는 재 테스트(Re testing)에 활용하여 자동화를 수행할 뿐이다. 많은 경우 웹 어플리케이션의 사용자 인터페이스는 수시로 변경되는데 그때마다 스크립트를 작성하는 것은 사용자에게 많은 부담을 줄 것이다. 본 논문의 경우 사용자 인터페이스가 변경되더라도 테스트 드라이버 자체를 자동으로 다시 생성하기 때문에 변경에 유연하게 대처할 수 있다는 장점이 있다. 마지막으로 ③부분은 실제 테스트 대상 페이지가 내부프레임(Inner Frame)에 포함되는 부분이다. 테스트 대상 페이지를 내부 프레임을 이용해 드라이버에 포함하는 이유는 스크립트 언어를 이용해 해당 페이지를 제어하기 위함이며, 클라이언트 측 페이지의 기능 테스트를 브라우저 기반으로 수행하기 위해서다.

3.3.3 자동 배치

배치의 자동화는 테스트 드라이버와 테스트 데이터들을 실제 서버에 설치하여 사용자가 테스트를 수행할 수 있도록 환경을 구축하는 것을 말한다. 서버 측 페이지의 기능 테스트와 클라이언트 측 페이지의 사용자 인터페이스 테스트의 실행 결과는 HTML 스트림이다. 이는 테스트 대상 페이지의 기능을 요청하였을 때 사용자에게 전달되는 페이지이며, 해당 어플리케이션이 배치된 서버 내에 테스트 드라이버와 관련 파일들이 배치되어야 정확한 결과를 확인할 수 있다. 클라이언트 측 페이지의 기능 테스트의 경우 테스트 드라이버가 서버 내에 배치될 필요는 없다. 그러나 본 연구에서는 테스트 드라이버가 대상 페이지를 포함하고 있고, 그것이 브라우저 상에서 동작하는 형태이므로 테스트 드라이버와 관련 파일들을 서버 내부에 배치하여야 정확한 결과를 확인할 수 있다.

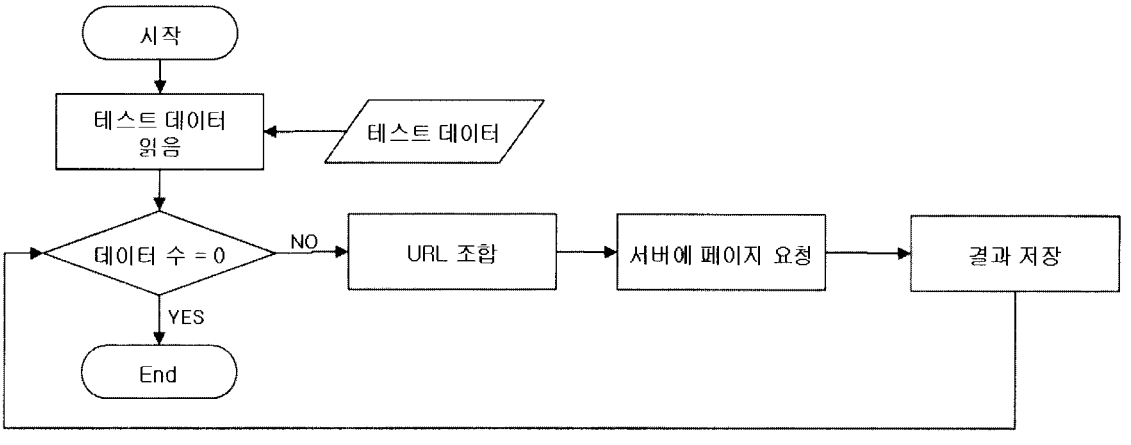
기존의 연구들은 테스트 도구를 클라이언트 측에 두고 테스트 실행을 한 다음, 서버에서 응답으로 전달되는 HTML 스트림을 저장하여 비교하는 방식으로 결과를 판단하였다. 그러나 HTML 스트림은 텍스트 형태이기 때문에 그것을 눈으로 직접 보고 정확하게 수행되었는지 판단하는 것은 쉬운 일이 아니다. 본 연구에서는 실행의 결과를 서버에 저장하고, 그것을 브라우저를 통해 관찰하고 판단할 수 있도록 하는 방식으로 바꾸었다. 테스트 실행 결과를 서버 측에서 브라우저에 띄워야 하는 또 다른 이유는 응답으로 전달되어 저장된 HTML 스트림을 클라이언트 측 브라우저 상에서 띄운다면 그림과 같은 요소들은 서버에 존재하기 때문에 정확한 모습이 나타나지 않는 이른바 '화면 깨짐' 현상이 발생하기 때문이다. 따라서 웹 어플리케이션의 테스트 결과는 서버 내에 저장해야 하며, 이를 위해 테스트 드라이버와 테스트 데이터를 서버 내에 배치해야 한다. 그러나 자동으로 생성된 테스트 관련 파일들에 대해 사용자가 모두 파악하는 데는 많은 노력이 요구되며, 이를 서버 내에 직접 배치하거나 테스트 수행 후 테스트 관련 파일을 직접 제거하기란 쉽지 않다. 따라서 배치의 자동화는 필수적이다. 이를 위해 본 연구에서는 테스트 자동화를 위해 테스트에 관련된 파일(테스트 드라이버, 테스트 데이터)을 서버에 자동으로 배치하고, 테스트 수행 후에 관련 파일을 서버에서 삭제하는 기능을 구현하였다.

3.4 테스트 실행 단계(Test Executing Stage)

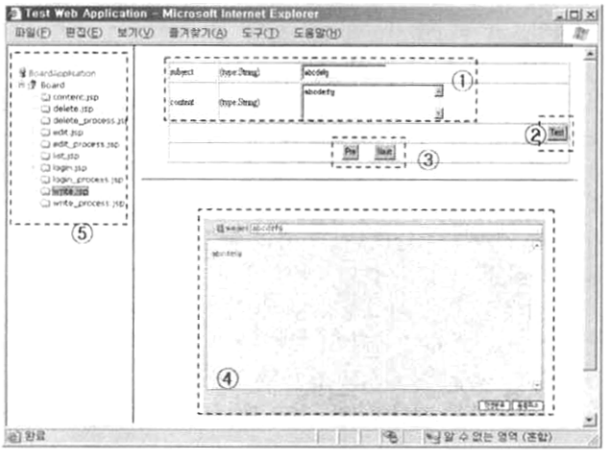
테스트 준비 단계를 통해 생성되는 테스트 드라이버와 테스트 데이터를 이용해 테스트를 자동으로 수행하는 단계이다. 앞서 설명한 바와 같이 테스트 드라이버는 세 가지 종류로 나뉘어 진다.

서버 측 페이지의 기능 테스트와 클라이언트 측 페이지의 사용자 인터페이스 테스트를 위한 드라이버는 태스트 대상 페이지와 파라메터 값 쌍의 조합을 이용해 URL을 생성하여 서버에 요청하고, 그 결과를 저장하는 방식으로 동작한다 (그림 13 참조). 최초에 사용자가 테스트 대상을 선택한 후 수행시키면, 테스트는 데이터의 수만큼 수행된다. 테스트 수행 후 결과 판정은 테스트 드라이버에 의해 저장된 실행 결과를 보고 사용자가 판단한다.

클라이언트 측 페이지의 기능 테스트는 브라우저 상에 대상 페이지가 로드된 상태에서 테스트가 수행되기 때문에 위의 두 테스트 방법에 비해 사용자의 개입을 더 많이 요구한다. 즉 클라이언트 측 페이지의 사용자 입력 제어 기능은 웹 브라우저를 통해서 동작하며, 실행 결과 또한 브라우저에 즉각적으로 나타난 후 사라지기 때문이다.



(그림 13) 서버 페이지의 기능 테스트 및 클라이언트 페이지의 UI 테스트를 위한 테스트 드라이버 동작

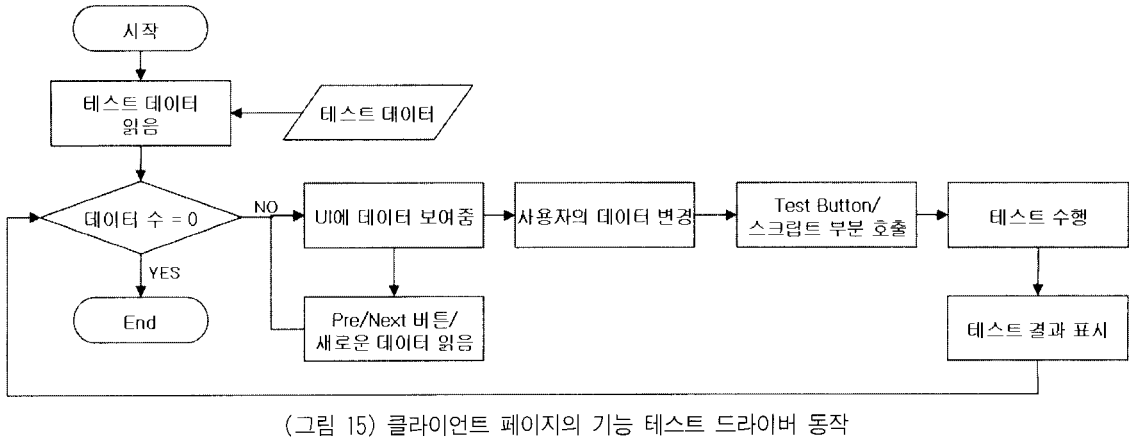


(그림 14) 클라이언트 페이지의 기능 테스트 드라이버

클라이언트 측 페이지의 사용자 입력 제어 기능 테스트를 위한 테스트 수행은 (그림 14)와 같은 인터페이스를 이용하며 (그림 15)의 방식으로 동작한다. (그림 14)의 테스트 드라이버에서 ①은 테스트 데이터 파일로부터 읽어 들인 데이터를 사용자에게 보여주는 부분이다. 이 부분은 테스트 수행 시 사용자가 직접 데이터를 수정할 수 있게 함으로써 테스트의 효율을 높일 수 있다. ②는 내부의 자바스크립트를 호출하여 내부 프레임에 포함되는 테스트 대상 페이지를 실행시키는 버튼이다. ③은 여러 테스트 데이터를 테스트할 때 다음 데이터를 읽어 들일 수 있도록 지원하는 부분이다. ④는 실제 테스트 대상 페이지가 내부 프레임에 포함되는 부분이다. ⑤는 테스트 대상 어플리케이션의 전체 구조를 사용자에게 보여주고 선택하여 테스트를 수행할 수 있도록 지원하는 부분이다. 이 방법에서는 스크립트 언어를 이용해 사용자의 입력을 시뮬레이션하고, 그 실행 결과를 통해 의도했던 기능이 제대로 수행되는지 판단한다.

3.5 테스트 결과 분석 단계(Test Result Analyzing Stage)

웹 어플리케이션의 테스트 결과는 HTML 스트림 형태이다. 따라서 테스트 오라클을 자동으로 생성하는 것은 어려운 일이다. 현재 웹 어플리케이션의 기능 테스트를 위한 몇몇 연구들이 수행되어 있지만, 테스트 오라클을 자동으로 생성하는 것에 관한 연구는 없다. 몇몇 연구들이 정상적인 상황에서의 테스트 결과를 테스트 오라클로 이용하여 테스트 결과와의 비교를 통해 결과 분석을 수행하고 있다. 그렇지만 이는 단순히 재 테스트(retest)에 이용할 수 있을 지 모르지만 사용자 인터페이스의 변경이나 HTML 구조의 변경이 이루어지는 회귀 테스트(regression test)의 경우 적용이 불가능하다. 따라서 본 논문에서 테스트 결과는 사용자



(그림 15) 클라이언트 페이지의 기능 테스트 드라이버 동작

또는 테스터가 직접 보고 판단하며, 그 편의를 위해 에러 상황과 정상적인 상황을 구분하여 어떤 상황이 에러를 유발하는지 확인 할 수 있도록 지원한다.

4. 웹 어플리케이션 테스트 자동화 시스템 프로토타입 및 사례

본 논문에서는 (그림 16)과 같이 웹 어플리케이션 기능 테스트 자동화 시스템의 프로토타입을 설계 및 구현하였다.

시스템은 (그림 16)에서 보는 바와 같이 페이지 관계 분석 모듈, 정보 추출모듈, 테스트 드라이버 생성 모듈, 테스트 데이터 생성 모듈, 배치 관리자로 구성되며 다음과 같이 동작한다. 테스트 대상 어플리케이션이 웹 서버에 배치되어 있다는 것을 가정한다.

① 페이지 관계 분석 모듈은 웹 서버에서 소스코드 파일 리스트를 읽고, 각 페이지의 HTML코드를 읽어 들인다. 각각의 정보를 바탕으로 파일 간의 관계를 테이블로 저장한다.

② 정보 추출 모듈은 ①에서 만들어진 페이지의 관계와 각 파일의 소스 코드, HTML 코드를 이용해 테스트에 필요한 정보를 추출한다.

③ 테스트 드라이버 생성 모듈과 테스트 데이터 생성 모듈은 ②에서 추출된 정보를 이용하여 테스트 드라이버와 테스트 데이터를 생성한다. 두 가지 작업은 병행하여 수행된다.

④ 배치 관리자 모듈은 생성된 데이터와 드라이버를 실제 서버 내에 배치하여 사용자가 브라우저를 통해 테스트를 수행할 수 있도록 환경을 구축한다.

⑤ 테스터는 배치된 테스트 드라이버와 테스트 데이터를 이용해 테스트를 수행한다.

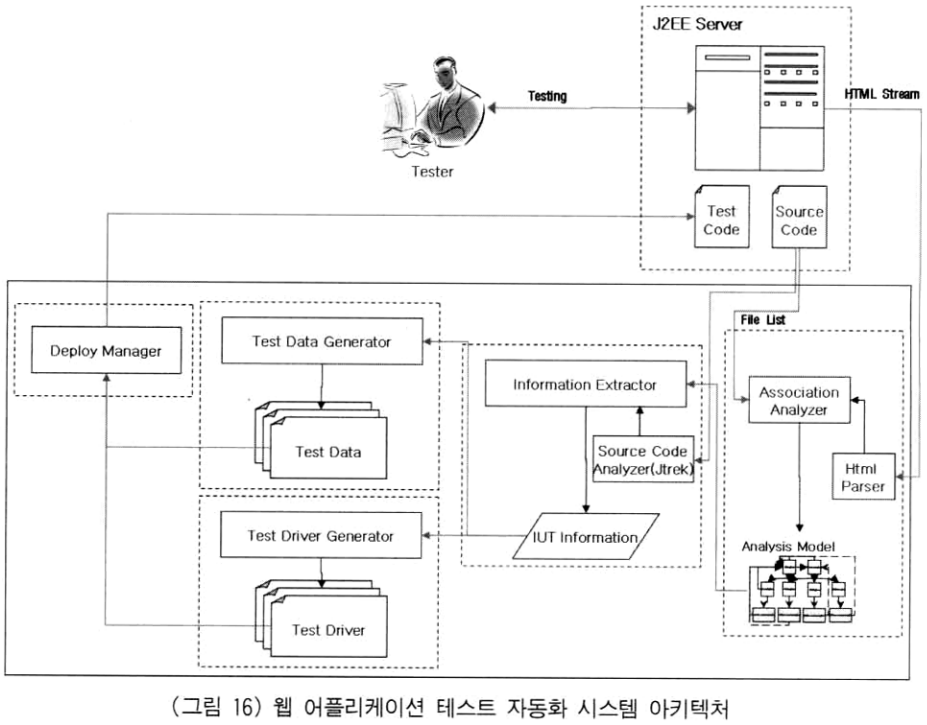
⑥ 테스트 수행 후 배치 관리자 모듈은 테스트와 관련된 모든 파일을 삭제하여 테스트 이전의 상태로 복구한다.

본 논문의 프로토타입 시스템을 이용하여 몇 가지 잘 알려진 예제 어플리케이션에 적용해 보았다. 대상 어플리케이션은 다음과 같으며, 사례 연구로 게시판 어플리케이션에 대해 설명한다.

게시판 어플리케이션에 대해 페이지 관계 분석 모듈은 (그림 7)과 같은 분석 모델을 생성하며, <표 3>과 같은 테스트 케이스도 추출한다. 표에서 테스트 케이스는 선행 페이지의 관계 정보로부터 추출된 것이다. 예를 들어 로그인 페이지는 게시물 목록 페이지의 링크를 통해 요청된다. 그리고 사용 패턴은 이러한 페이지들 간의 관계 타입을 나타낸다. 마지막으로 표의 테스트 케이스 항목은 대상 페이지의 테스트를 수행할 때 넘겨져야 하는 파라메터의 종류를 나타내는 것으로 실제 URL에서 파라메터 전달 형태에 해당하며 각 파라메터는 &로 구분된다.

〈표 2〉 테스트 수행 어플리케이션

테스트 대상 어플리케이션	페이지 수	구현 언어
게시판	10	JSP
도서구매	18	JSP
인터넷 설문조사	13	JSP



(그림 16) 웹 어플리케이션 테스트 자동화 시스템 아키텍처

〈표 3〉 게시판 어플리케이션의 테스트 케이스

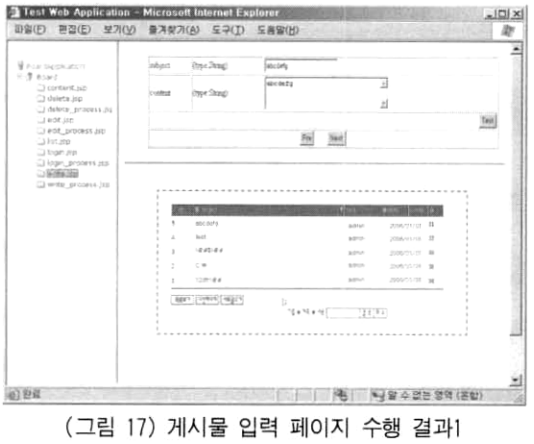
테스트 대상 페이지	선행 페이지	사용 패턴	테스트 케이스 항목
login.jsp	list.jsp	<a>	
login_process.jsp	login.jsp	<form>	ID&Password
list.jsp	login_process.jsp	<redirect>	
list.jsp	list.jsp	<a>	page_id&searchType&searchValue
list.jsp	list.jsp	<form>	searchType&searchValue
list.jsp	content.jsp	<a>	uid&page_id&searchType&searchValue
content.jsp	list.jsp	<a>	page_id&searchType&searchValue
.....	.....		.....
write.jsp	list.jsp	<a>	page_id&searchValue&searchType
write_process.jsp	write.jsp	<form>	subject&content

〈표 4〉 게시판 어플리케이션의 데이터 타입 추출

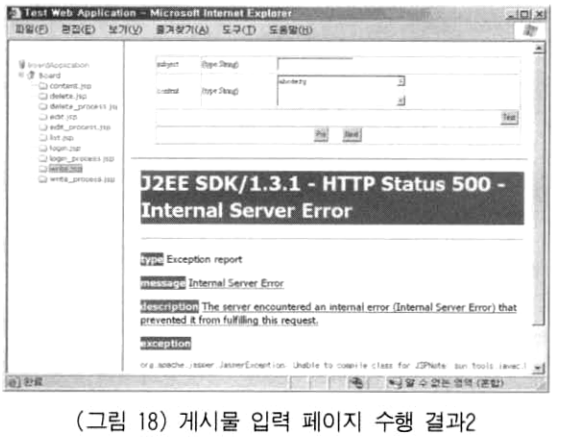
테스트 대상 페이지	선행 페이지	사용 패턴	파라메터 이름	파라메터 타입
write.jsp	list.jsp	<a>	page_id	int
			searchValue	String
			searchType	String
write_process.jsp	write.jsp	<form>	subject	String
			content	String

〈표 5〉 게시판 어플리케이션의 테스트 데이터

테스트 대상 페이지	파라메터	테스트 데이터						
		1	2	3	4	5	.....	N
write.jsp	page_id	1000	10	1	0	1	.....	1000
	searchValue	abcde…	!@#$%...	가나다…	12345…	D0 CF..	.....	NULL
	searchType	./test.jsp	12345…	NULL	D0 CF	!@#$%...	.....	abcde…
write_process.jsp	subject	abcde…	NULL	D0 CF …	./test.jsp	가나다..	.....	12345…
	content	D0 CF …	가나다…	12345…	NULL	abcde..	.....	abcde…



(그림 17) 게시물 입력 페이지 수행 결과1



(그림 18) 게시물 입력 페이지 수행 결과2

다음 <표 4>는 정보 추출 모듈에 의해 추출된 게시판 등록페이지에 대한 정보와 게시물 등록 처리 페이지에 대한 정보이다. 게시판 등록 페이지는 클라이언트 측 페이지이고, 게시물 등록 처리 페이지는 서버 측 페이지이다.

<표 5>는 위 <표 4>의 각각 페이지에 대해 생성한 테스트 데이터의 일부를 보여주며, 실제 테스트 수행 시 각 태

〈표 6〉 테스트 수행 결과

테스트 대상 어플리케이션	페이지 수	테스트 평균 수행 횟수	테스트된 페이지	Error
게시판	10	12/page	10	0 Errors
도서구매	18	25/page	18	0 Errors
인터넷 설문조사	13	18/page	13	0 Errors

스트 데이터를 이용하였다.

게시물 등록 페이지에 대한 테스트 드라이버는 (그림 14)와 같은 형태로 구현되며 (그림 15)와 같은 방법으로 수행된다. 수행 결과는 (그림 17)과 같이 게시물 목록 페이지가 내부 프레임에 포함되어 사용자에게 보여지게 되며, 오류가 발생할 경우 (그림 18)과 같은 결과를 보여준다.

본 논문에서는 <표 2>의 테스트 대상 어플리케이션에 대해 프로토타입 시스템을 적용하였지만 각 어플리케이션이 이미 안정된 시스템이기 때문에 테스트 데이터에 의한 오류 이외엔 어떠한 오류도 발견되지 않았다. 테스트 데이터에 의해 발생하는 오류를 배제하였기 때문에 <표 6>에서는 발견된 오류의 개수가 전부 0으로 표현되어 있다. 향후 본 논문의 시스템을 개발중인 어플리케이션에 적용할 예정이고, 그럴 경우 다양한 오류를 검출할 수 있을 것으로 기대한다. 다음 <표 6>은 위 세 가지 어플리케이션에 대한 테스트 수행 결과이다.

5. 결론 및 향후 연구 방향

인터넷의 급속한 성장과 웹 기술의 성장은 웹 어플리케이션을 점점 더 복잡하게 하였다. 또한 다양한 비즈니스 요구사항 중 웹 어플리케이션의 품질 및 신뢰성에 대한 요구가 증가하고 있다. 이러한 요구사항을 만족하기 위해서는 웹 어플리케이션 개발 과정에서 적절한 테스트는 필수적이다. 그러나 웹 어플리케이션의 기능 테스트에 관한 노력은 미흡하며, 자동화를 지원하는 도구 또한 적다.

본 논문에서는 웹 어플리케이션의 테스트 자동화 방법을 제안하고 JSP 어플리케이션에 적용할 수 있는 프로토타입 시스템을 개발하였다. 웹 어플리케이션의 테스트를 자동화하기 위해 본 논문에서는 테스트 단계를 4단계 즉, 웹 어플리케이션 분석 단계, 테스트 준비 단계, 테스트 수행 단계, 테스트 결과 판단 단계로 구별하고, 이중에서 분석, 준비, 수행 단계의 테스트 자동화 방법을 제안하였다. 본 논문의 방법을 이용해 이미 프로토타입 시스템이 개발되었으며, 몇 가지 예제 프로그램에 적용하여 웹 어플리케이션의 테스트 자동화 사례를 보였다.

향후에 데이터 흐름 분석 및 스펙 기반의 신뢰성 높은 테스트 데이터 생성 방법에 관한 연구를 수행할 예정이다. 또한 테스트 수행을 간편화하도록 사용자 인터페이스의 개선을 수행할 예정이다. 그리고 테스트 결과 판단의 자동화를 위해 테스트 오라클의 자동생성에 관한 연구가 필요하다.

참 고 문 헌

[1] Rhonda Dibachi, "Testing e-commerce: Reducing your company's risk of doing business on the Web," Software Testing & Quality Engineering Magazine, pp.57-62, 1999.

[2] 한국정보통신기술협회, "소프트웨어 테스트 전문 기술-응용편," 2006.

[3] Edward Miller, "WebSite Testing," Software Research Inc., http://www.soft.com/eValid/Technology/White.Papers/website.testing.html.

[4] David Chenho Kung, "An Object Oriented Web Test Model for Testing Web Applications," 24th International Computer Software and Applications Conference, pp.537-542, 2000.

[5] Chien-Hung Liu Kung, "Structural testing of Web applications," Proceeding of 11th International Symposium on Software Reliability Engineering, pp.84-96, 2000.

[6] Sampath, S., "Composing a framework to automate testing of operational Web-based software," Proceeding of 20th IEEE International Conference on Software Maintenance, 2004.

[7] Giuseppe Antonio Di Lucca, "Testing Web Applications," Proceeding of the International Conference on Software Maintenance, pp.310-319, 2002.

[8] 권영호, 최은만, "웹 어플리케이션의 사용자 인터페이스 테스트 자동화 기법," 정보처리학회 논문지, pp. 293-300, 2003.

[9] Glenford J. Myers, "The Art of Software Testing 2nd Edition," John Wiley & Sons Inc., 2004.

국 승 학

e-mail : triple888@cnu.ac.kr
2004년 충남대학교 컴퓨터과학과 (이학사)
2006년 충남대학교 컴퓨터공학과 (공학석사)
2006년~현 재 충남대학교 컴퓨터공학과 박사과정
관심분야 : 소프트웨어 테스팅, 소프트웨어 품질관리, SOA, 웹 서비스

김 현 수

e-mail : hskim401@cnu.ac.kr
1988년 서울대학교 계산통계학과 (이학사)
1991년 한국과학기술원 전산학과 (공학석사)
1995년 한국과학기술원 전산학과 (공학박사)
1995년~1995년 한국전자통신연구원 Post Doc.
1996년~2001년 금오공과대학 조교수
1999년~2000년 Colorado State University 방문교수
2001년~현 재 충남대학교 전기정보통신공학부 부교수
관심분야 : 소프트웨어 테스팅, 소프트웨어 재공학, 컴포넌트 마이닝, 컴포넌트 테스팅, SOA