웹 어플리케이션의 사용자 인터페이스 테스트 자동화 기법

권 영 호[†] · 최 은 만[††]

요 약

인터넷의 급속한 성장과 웹 관련 기술이 더 복잡해지면서 웹 응용 소프트웨어의 품질과 신뢰성이 중요하게 되었다. 웹 기반 소프트웨어는 전 세계에 흩어져 있는 수많은 사용자들을 대상으로 한다는 면에서 품질과 시험 방법 또한 중요하다. 이 논문은 브라우저 객체를 이용하여 HTML 웹 페이지의 사용자 입력 부분에 대한 테스트 케이스를 자동적으로 실행할 수 있는 방안을 제시하고 JavaScript로 매핑하여 자동화가 가능함을 보였다. 스크립트 작성기를 만들면 JavaScript를 작성하는 오버헤드를 줄일수도 있다. 작성된 테스트 스크립트는 웹 기반 소프트웨어의 리그레션 테스트에 반복적으로 사용할 수 있다.

Automation Technique of Testing User Interface of Web Application

Youngho Kwon[†] · Eun Man Choi[††]

ABSTRACT

As Internet has grown rapidly and been more complex by technology in connection with Web and requirement of business, quality and reliability of Web application are getting important. It is necessary to study about testing method along with design technique specially in Web application. This paper explains automation method of user interface test to make test cases about user input form with HTML pages using by built-in browser objects. Examples shows the possibility of testing automation with JavaScript objects get mapped. Overhead of writing JavaScript can be reduced by making script generator. Generated test scripts are repeatedly used in regression testing Web-based application.

키워드 : 웹 테스트(Web test), 소프트웨어 테스트(software tesing), 사용자 인터페이스(user interface), 웹 응용(Web application)

1. 서 론

인터넷과 웹의 급속한 성장은 비즈니스; 상거래, 산업, 은행, 재정, 엔터테인먼트, 교육 부분 등 우리의 일상 여러 면을 변하게 하고 있다. 웹 관련 기술의 발전과 함께 다양한 비즈니스 요구사항이 전통적인 어플리케이션을 웹 환경으로 전환하게 하고 있다[1]. 웹은 광범위한 사용이라는 특성 때문에 좋은 품질과 신뢰성이 필수적이나 컨텐트의 즉시성(immediacy), 즉 잦은 변경 때문에 좋은 품질을 유지하기가 쉽지 않다. 전통적인 클라이언트-서버 어플리케이션에 비해 웹 어플리케이션이 점점 더 복잡해지고 있으며, 다양한 웹 기술을 가지고 테스팅과 품질 관리를 하기가 더욱 어려워지고 있다[2-4].

웹 기반 시스템에 대한 효율적인 개발과 유지보수 및 확장에서 좋은 품질을 유지하기 위해서 시스템 개발의 생명주기에 따라 적절한 테스팅 방법이 필수적이다. 개발 단계에서는 개발중인 시스템에 존재할 수 있는 병목현상과 시스템의 성능저하 같은 문제점을 미리 예측함으로써 즉각적인 문제 해결이 필요하다. 테스팅 단계에서는 가상 사용자 시뮬레이션을 통하여 실제 시스템 운영 시에 발생할 가능성이 있는 문제점들을 시험할 수 있어야 한다. 또한 시스템을 배포하여 운영하는 단계에서는 운영중인 시스템에 대한 모니터링을 통하여 발생 가능한 문제점을 예측하고 미리 대처하거나 사용자 수의 증가에 따른 시스템 확장 계획 수립에 적용할 수 있어야 한다.

웹 어플리케이션 테스팅의 중요도에 비해 연구는 부족하다. 현재 개발 현장에서 웹 테스팅에 사용하고 있는 기법은 대부분 전통적인 기법(예를 들면 기능테스트, DB 테스트, 보안, 성능 테스트 등)을 사용하고 있고 십여가지의 테스팅 도구들을 출시하고 있다[5]. 하지만 웹 테스팅의 핵심이라고 할 수 있는 가상 시뮬레이션을 위한 자동화된 테스트 스크립트 작성에 대한 부분은 아직 초보적인 단계에 불과하다. 테스트 스크립트를 자동으로 작성하는 방법으로 웹 브라우저상의 마우스 클릭이나 키보드 입력과 같은 사용자 행동을 레코딩하는 방법, 로그파일을 분석하는 방법, 디렉토리 구조 정보를 이용하는 방법 등이 제시되고 있지만 실제 상황과 유사한 테스트 스크립트 작성과는 거리가 멀다. 특히, 사용자 입력을

† 준 회 원 : DTV Interactive
†† 정 회 원 : 동국대학교 컴퓨터공학과 교수
논문접수 : 2002년 2월 18일, 심사완료 : 2002년 11월 13일

필요로 하는 동적 페이지인 경우는 이러한 기능이 극히 제한적이거나 기능을 전혀 제공해 주지 않고 있다.

따라서 보다 발전된 웹 테스팅을 위해서는 사용자 또는 사용자 환경에 대한 패턴 및 경향 분석을 통한 실제 상황에 근접한 테스트 스크립트 작성에 대한 연구와 사용자 입력을 처리하는 동적 페이지에 대한 테스트 스크립트 작성에 대한 연구가 절실히 필요하다. 이 논문에서는 증가하는 웹 어플리케이션을 테스트하는 다양한 방법 중에서 사용자 인터페이스 테스팅을 자동화하는 방안을 제안한다. 여기서는 대부분의 웹 어플리케이션이 웹 브라우저 기반의 사용자 인터페이스를 사용하므로 사용자의 입력 부분과 마우스의 누름에 대한 반응을 테스트의 주된 관심사로 생각한다. 따라서 사용자 입력 폼(Form), 즉 텍스트 필드, 라디오 버튼, 체크박스, 선택박스 등에 대한 테스트 자동화 기법을 제시한다.

2. 웹 기반 테스트에 관한 연구

웹 기반 어플리케이션은 다양한 컴포넌트로 구성된다. 클라이언트에는 웹 브라우저, 스크립트, 플러그 인 컴포넌트 등 사용자 서비스 컴포넌트들이 있고, 서버에는 비즈니스 로직이 있는 응용 서버 컴포넌트가 있으며, 데이터 저장을 위하여 데이터 서비스 컴포넌트가 있다. 전통적인 테스트 방법을 따른다면 각 컴포넌트에 대한 단위시험을 한 후 통합시험을 생각할 수 있다. 그러나 웹 기반 시스템은 데이터베이스, CGI, Java, ASP, JSP, PHP, ActiveX, EJB 등 단일 개발에 대해 여러 가지의 언어와 기술이 통합적으로 사용되어 컴포넌트를 이루는 정확한 단위로 분할하기가 어려운 점이 있다.

웹 어플리케이션을 테스트하는 방법은 웹 페이지가 정적이냐 동적이냐에 따라서 달라질 수 있다. 정적인 경우는 HTML 구문 검사와 링크 검사로 가능하지만, 동적으로 서버 측에서 생성할 경우에는 통합 테스트가 필요하다[16]. 이제까지 진행된 웹 기반 시스템에 대한 테스팅을 위한 연구로는 전체 웹 사이트를 테스트 가능한 컴포넌트(또는 객체)로 구분하고 이를 반복적으로 테스트함으로써 전체 웹사이트를 테스트하는 방법[8]과 접근로그를 이용하여 통계학적 테스팅을 위한 모델을 만들고 에러로그를 분석하여 신뢰성을 검증하는 연구[9], 웹 어플리케이션을 객체, 행위, 구조적 관점에서 분석하는 방법[10], 웹 페이지를 구성하는 응용 컴포넌트들의 구조를 파악하고 각 페이지를 구동시키는 액션들을 찾아내어 상태기반의 테스트 데이터를 찾아내는 방법[15, 16] 등이 있다.

그러나 위에 열거한 방법들은 실용적인 관점에서 보면 많은 문제점들을 안고 있다. 그 문제점을 살펴보면,

① 웹어플리케이션을 구성하는 컴포넌트로의 분할이 쉽지 않다.
② 빈번히 일어나는 변경으로[7] 테스트 노력이 많이 들고 반복적인 작업이 많다.
③ 정적 분석으로 테스트 드라이버의 작성이 가능하나 대부분 네비게이션 링크, 철자법, HTML 호환 오류 등을 찾아내기 위한 목적으로만 사용되고 있다.

웹 기반 소프트웨어는 단순한 HTML로 구성된 정적인 웹페이지 테스트에서 적용한 정적 분석보다는 사용자 인터액션에 의한 반응과 기능 위주의 통합시험이 필요하다.

사용자 인터액션에 초점을 두고 있는 구현분야에 관한 다른 기술과 비교한다면 다음과 같다. 많이 사용되는 방법인 상태기반 테스트 방법은 웹의 상태 변화를 일으키는 링크와 사용자 입력을 모델링하여 입력 순서에 따른 상태 변화를 파악하고 이를 WinRunner와 같은 도구를 이용하여 테스트하는 방법이다[17]. 이 방법은 사용자 인터페이스를 검증하는 입력 및 웹 구성 요소들을 일일이 수동으로 파악하고 사용자 액션과 입력 필드사이의 관계를 스크립트로 표현하고 이를 기초로 테스트 도구가 해당되는 부분을 구동하여 출력 결과를 검토하는 방법이므로 테스트에 소요되는 시간과 노력이 크다.

반면 이 연구에서는 사용자 인터페이스 부분을 웹 문서에서 파악하여 이를 구동할 수 있는 스크립트를 자동으로 만들 수 있는 방안을 연구하였다. 즉 사용자 인터페이스 위주의 동적인 테스트를 구동하기 위한 JavaScript의 작성 방법 및 이의 생성 자동화 방안에 대하여 연구하였다.

3. UI 중심 웹 페이지 테스트 기법

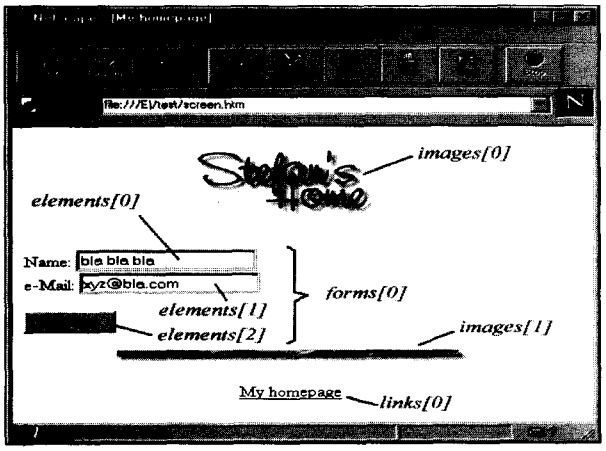
3.1 HTML 문서의 브라우저 표현

기술의 발전과 함께 웹 사용자 인터페이스는 사용하기 쉬운 GUI와 명령어 입력 인터페이스(command-line interface)가 되었다. WUI는 HTML로 이루어져 있으므로 먼저 HTML 문서가 웹 브라우저에서 어떻게 표현되는지 알아보아야 한다.

HTML 문서가 웹 브라우저에 로딩될 때, 그 문서의 서로 다른 구성요소는 자바스크립트 객체를 생성한다. 예를 들면, 하나의 이미지 객체는 HTML 문서의 <img> 태그로부터 생성된다. 이렇게 생성된 객체는 속성과 메소드를 가진다. 그 속성은 HTML 구성요소의 속성으로 표시되고, 메소드는 호출될 때 객체에 대해서 적절한 명령을 수행한다. 텍스트 객체의 속성 중 하나는 폼에서 입력하는 값을 표시하는 ***value*** 이다. ***focus*** 메소드는 호출될 때, 브라우저 화면에서 텍스트 필드에 포커스를 맞추게 된다.

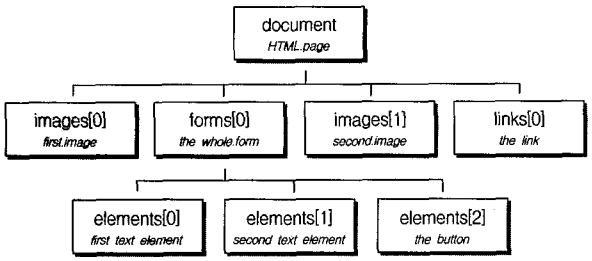
이와 비슷하게 브라우저는 객체를 생성하는데, 그때 HTML 페이지 자체의 구조를 반영하는 계층구조로써 초기화와 정렬을 하게 된다. 브라우저 내장 객체는 HTML 페이지의 구조를 나타내는 계층구조를 하고 있다. 이러한 HTML의 모든 구성요소들은 자바스크립트 객체로 매핑이 된다. 자바스크립트는 웹 페이지 상에 나타나는 모든 구성요소를 하나의 계층구조로 조직화한다[11]. 자바스크립트를 이용하면 쉽

게 HTML 문서 객체를 조작할 수 있다.



(그림 1) 브라우저의 HTML 문서 표현[11]

HTML 객체는 (그림 1)에서 보는 것 처럼 웹 브라우저 상에 표현된다. 자바스크립트의 관점에서 보면, 브라우저 윈도우는 하나의 윈도우 객체이다. 윈도우 객체에는 메뉴바, 툴바, 상태바와 같은 여러 가지 요소들이 포함된다. 하나의 윈도우 안에서 하나의 HTML 문서를 불러들일 수 있는데 이때 로딩되는 웹 문서가 하나의 도큐먼트 객체에 해당된다. (그림 2)는 (그림 1)의 HTML 문서 객체의 계층구조를 보여주고 있다.



(그림 2) HTML 문서 객체의 계층구조[11]

3.2 HTML 객체에 접근하는 방법

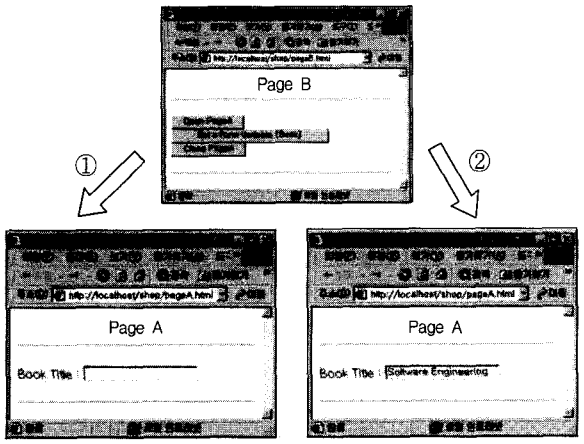
여기서 HTML 문서 내의 서로 다른 객체에 대한 정보와 그 객체를 조작하는 방법에 대해 알아보면, 먼저 해당 객체에 접근하는 방법을 알아야 하는데 (그림 2)의 계층구조에서 각 객체에 붙여진 이름을 볼 수 있다. HTML 문서의 첫 번째 이미지에 접근하려면 계층구조의 맨 위의 도큐먼트 객체에서 시작해서 첫 번째 이미지는 images[0]를 통해 참조가 된다. 결국 자바스크립트를 이용하면 document.images[0]와 같이 접근할 수 있다.

입력란에 어떤 문자열이 입력되었는지 알려면 입력 객체의 속성과 메소드를 보면 된다. 입력 텍스트 객체의 속성은 value이다. 객체가 많은 문서에서는 계층구조의 순서상의 숫자로 구분한다면 혼동이 올 수 있을 것이다. 이 문제를 피하기 위해 객체의 이름으로 대신 사용할 수 있다.

```
<form name = "myForm">
Name : <input type = "text" name = "myText" value = " "><br>
...
```

(그림 3) HTML 입력 폼 예

(그림 3)의 예제에서 forms[0]는 'myForm'으로 불려진다. 여기서 name = document.forms[0].elements[0].value ; 는 name = document.myForm.myText.value ; 로 쓸 수 있다.



(그림 4) 다른 HTML 문서 객체의 접근

(그림 4)에서 페이지 A와 B와 같이 웹 서버에서 요청된 HTML 문서가 있다. 페이지 A는 "Book"이라는 이름의 텍스트 객체를 포함하고 있는 폼(form)을 가진 간단한 HTML 문서이다. 페이지 B는 페이지 A 객체에 접근하기 위해 자바스크립트 함수를 가지고 있다. 페이지 B는 세 개의 버튼을 가지고 있는데, 첫 번째 버튼(Open Page A)은 새로운 브라우저 창에 페이지 A를 로딩한다((그림 4)에서 ①번). 두 번째 버튼(Set a Form Variable[Book])은 페이지 A 객체 계층구조에 접근하여 텍스트 객체의 value를 "Software Engineering"으로 설정한다((그림 4)에서 ②번). 세 번째 버튼은 페이지 A를 닫는다. 여기서 페이지 B는 페이지 A의 객체에 접근하여 사용자 입력 필드에 대해서 "Software Engineering"이라고 입력시킨다. 이렇게 자바스크립트를 이용하여 HTML 문서의 사용자 입력 폼에 접근 가능하다. (그림 5)와 (그림 6)은 페이지 A와 B에 대한 소스 코드이다.

```
<html>
<body>
  <center>
  <font size = "5"> Page A </font><br><hr>
  </center>
  <form method = "post">
    Book Title : <input type = "text" name = "book">
  </form>
  <hr>
</body>
</html>
```

(그림 5) 페이지 A의 소스 코드

```
<html>
<head>
   <script language = "Javascript">
       var Win ;
       // ----------------------------------------
       // Open 'Page A'
       // ----------------------------------------
       function init( ) {
            Win = window.open("http://localhost/pageA.html") ;
       }
       // ----------------------------------------
       // Modify 'Page A'
       // ----------------------------------------
       function modifyFormA() {
        // Set a value to form variable - Book
        Win.document.forms[0].book.value = "Software
                                           Engineering" ;
       }
       // ----------------------------------------
       // Close 'Page A'
       // ----------------------------------------
       function closeA( ) { Win.close( ) ; }
   </script>
</head>
<body >
  <center>
  <font size = 5> Page B </font><br><hr>
  </center>
  <form name = "formB" action = " ">
    <input type = "button" value = "Open PageA"
      onClick = "init( ) ; " ><br>
    <input type = "button" value = "Set a Form Variable [Book]"
      onClick = "modifyFormA( ) ; " ><br>
    <input type = "button" value = "Close PageA"
      onClick = "closeA( ) ; " ><br>
  </form>
  <hr>
</body>
</html>
```

(그림 6) 페이지 B의 소스 코드

3.3 HTML의 체계적인 구문분석

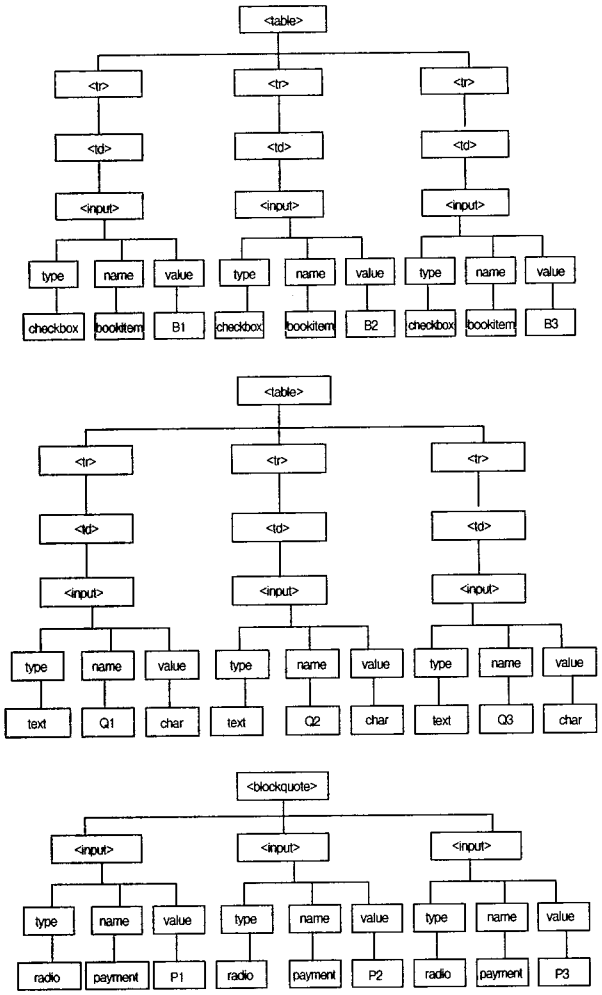
자동화 프로그램의 제작에 앞서, 프로그래밍에 필요한 요소를 HTML 문서에서 추출해 내는 과정은 매우 중요하다고 할 수 있다.

도큐먼트 객체 안에서 폼에 대한 요소들을 필요로 한다. 폼에 대한 요소로는 텍스트 필드, 체크박스, 라디오 버튼, 문서 입력 창, 선택 입력 양식 등이 있다.

다음은 HTML을 XML의 DOM으로 표현한 것이다. DOM은 HTML과 XML 문서를 위한 API이다. 또한, 객체지향 디자인이며, 인터페이스와 객체들의 집합을 나타낸다. 특성으로는 HTML과 XML 문서 안에 있는 어떤 것도 접근, 갱신, 삭제, 추가가 가능하며, 어떤 컴퓨터 환경에서도 표현이 가능하다. 하지만, 단지 Functions과 identity를 가지는 객체를 표현할 뿐 데이터 구조를 표현하지는 않는다.

(그림 7)은 DOM을 사용하여 HTML 도큐먼트를 트리로 표현한 것이다. DOM은 전후/위 아래 어느 곳으로든 자유롭게 이동할 수 있고, 미리 삽입해둔, 도큐넌트의 특정부분으로 곧장 찾아갈 수 있다.

DOM은 문서의 요소와 컨텐트를 애플리케이션이나 프로그래밍 언어가 쉽게 액세스하고 조작할 수 있도록 계층적인 트리의 객체로서 나타낸다.



(그림 7) HTML 문서의 계층적 표현

3.4 웹 사용자 인터페이스 테스팅 자동화

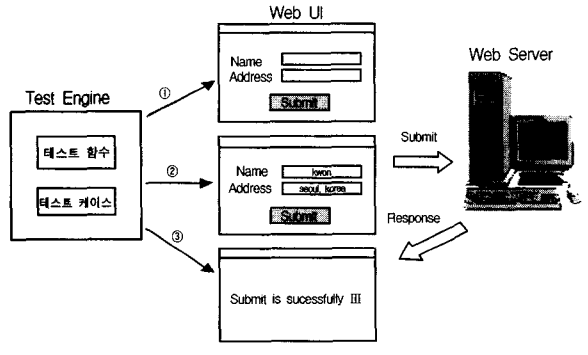
웹 어플리케이션의 사용자는 웹 브라우저의 인터페이스로 웹 서버와 상호작용을 한다. 웹 테스팅이 자동화되지 않은 경우에는 어플리케이션의 단위 테스트 위주의 수동 테스팅(manual testing)을 하게 되는데, 이는 많은 시간적인 노력과 테스터가 오류를 범할 수 있다. 명심할 것은 테스팅을 자동화한다고 해서 모든 문제가 해결된다는 것이 아니라는 것이다. 테스팅을 자동화하는 과정도 숙련된 지식과 노력이 필요하며 테스트 스크립트를 작성하는 과정 또한 프로그래밍의 과정이라는 사실을 염두에 두고 있어야 한다.

이 논문에서는 대부분의 웹 어플리케이션이 웹 브라우저 기반의 사용자 인터페이스를 사용하므로 사용자의 입력 부분과 마우스의 누름에 대한 반응을 테스트의 주된 관심사로 생각한다. 따라서 사용자 입력 폼, 즉 텍스트 필드, 라디오 버튼, 체크박스, 선택박스 등에 대한 테스트 자동화 기법을 제시한다.

야후 같은 데이터베이스 검색 시나리오를 생각해 보자.

① 사용자는 검색 버튼을 누른다.
② 키워드 입력 양식으로 돌아간다.
③ 사용자는 키워드(검색어)를 넣고 전송한다.
④ 검색 결과를 본다.

이 경우에, HTML 문서에는 2개의 입력 이벤트(검색 버튼 누름과 키워드 입력)가 일어난다. 추가로 검색 결과로 많은 HTML 문서가 생성된다. 이것을 수동으로 테스트할 경우에 서로 다른 테스트 케이스(검색어)들을 가지고 매번 데이터베이스 검색 단계에서 수많은 반복을 하게 될 것이다. 그러나 자바스크립트로 이용해서 테스트 스크립트 만들면 사용자 이벤트에 대해 자동화된 테스팅을 수행할 수 있다.



(그림 8) 웹 인터페이스 테스팅 자동화 방법

(그림 8)은 테스팅 자동화 과정을 보여주고 있다. 이 과정의 테스트 시나리오를 설명하면,

① 입력 폼과 전송 버튼을 누르는 이벤트에 대해 자바스크립트를 이용해서 테스트 엔진을 만든다.
② (그림 8)①에서 테스트 함수가 테스트할 HTML 문서를 호출하고, 테스트 횟수를 설정한다.
③ (그림 8)②에서 텍스트 필드를 임의의 값으로 채운 후, 폼을 웹 서버로 전송(submit)한다.
④ 처리 결과를 가진 HTML 문서를 받을 때까지 기다린다(response).
⑤ (그림 8)③에서 처리 결과가 성공적인지 검사한다.

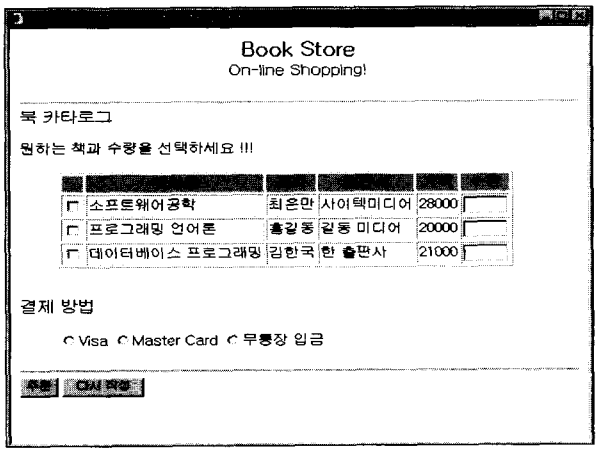
이렇게 테스트가 한 번 자동화되면, 다른 테스트 케이스를 가지고 여러 번 수행할 수 있게된다. 이 경우 테스트 함수는 매개변수로써 키워드 값을 가진다.

테스트 엔진은 테스팅을 자동화하는 핵심 컴포넌트이다. 테스트 함수는 웹 어플리케이션의 각각의 기능을 테스트하기 위해 사용된다. 테스트되어야 할 모든 특성들에 맞게 테스트 함수가 구성된다. 모든 테스트 함수는 자바스크립트로 작성된다. 각각의 테스트 함수는 어떤 특성에서 필요한 입력값을 제공해주고 그것의 객체를 조작해서 서버에 전송을 하고 결과를 기다리게 된다. 사용자의 브라우저 화면에서 결과를 확인하여 기대한 결과와 같지 않은 경우에, 테스팅을 중단하고 적절한 조치를 취해야 한다.

이렇게 만들어진 테스트 함수와 테스트 케이스는 테스트 후에 평가과정을 거쳐서 항상 정확한 결과가 나오도록 유지해야 한다.

4. 테스팅 자동화 사례

(그림 9)와 같은 온라인 서점에서 책을 주문하는 웹 어플리케이션이 있다고 하자. 이 온라인 서점은 고객의 주문을 웹 페이지를 통해 주문 받는다. 여기서 사용자가 주문하는 부분을 테스팅 자동화하는 과정을 예로 든다. 사용자는 주문할 책을 선택하고 수량과 결제방법을 선택해서 주문 버튼을 누른다. 웹 어플리케이션은 주문에 대한 결과를 사용자에게 다시 보여주며 사용자에게 주문한 정보가 맞는지를 확인하게 한다.



(그림 9) 온라인 서점 웹 어플리케이션

넷스케이프, 인터넷 익스플로러와 같은 웹 브라우저에서 자바스크립트가 모두 호환성이 있으므로 웹 브라우저는 인터넷 익스플로러 5를 사용하였으며, 웹 어플리케이션은 윈도우 2000 서버 플랫폼에서 IIS 웹 서버, PHP와 MySQL로 구성하였다.

(그림 10)에서 ①, ②, ③의 체크박스, 텍스트 필드, 라디오 버튼에 대한 속성에 테스트 케이스를 자동으로 입력해서 테스팅을 수행하게 된다.

그리고 테스트 엔진을 자바스크립트로 구성한다. (그림 13)과 같이 구성된 테스트 스크립트는 테스트 케이스를 생성하게 된다. (그림 13)에서 체크박스의 경우는 모두 true 값(④번), 텍스트 필드에는 난수를 발생(⑤), 라디오 버튼의 경우는 역시 임의의 값을 선택(⑥)하게 했다.

```
<html>
 <head>
   <title> Book Store </title>
  </head>
  <body bgcolor = "# ffffff">
  <center>
  <font size = 5> Book Store </font><br>
  <font size = 4> On-line Shopping! </font><br>
  </center>
  <br><hr>
  <font size = 4> 북 카타로그 <font>
```

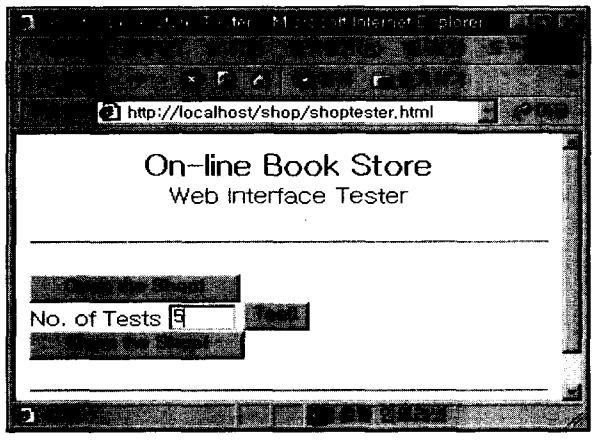
```
  <br><br>
<font size = 3> 원하는 책과 수량을 선택하세요 !!! </font>
<form method = "post" action = "http://localhost/shop/
                onlineShop.php">
  <blockquote>
  <!-- ------------------------------------------
   // Display Book items and their prices
    ------------------------------------------ -->
  <table border = 1>
    <tr bgcolor = skyblue><th><th> 책이름 <th> 저자 <th>
                출판사 <th> 가격 <th> 수량
    <tr><td><input type = "checkbox" name = "bookitem"
                value = "B1"> ← ①
        <td> 소프트웨어공학
        <td> 최은만
        <td> 사이텍미디어
        <td> 28000
  <td><input type = "text" size = 5 name = "Q1"> ← ②
    <tr><td><input type = "checkbox" name = "bookitem"
                value = "B2">
        <td> 프로그래밍 언어론
        <td> 홍길동
        <td> 길동 미디어
        <td> 20000
  <td><input type = "text" size = 5 name = "Q2">
    <tr><td><input type = "checkbox" name = "bookitem"
                value = "B3">
        <td> 데이터베이스 프로그래밍
        <td> 김한국
        <td> 한 출판사
        <td> 21000
        <td><input type = "text" size = 5 name = "Q3">

  </table>
  </blockquote>
  <br>
  <!-- ------------------------------------------
   // Display Payment Methods
   ------------------------------------------ -->
  <font size = 4> 결제 방법 <font>
   <blockquote>
     <input type = "radio" name = payment value = "P1">
          Visa ← ③
     <input type = "radio" name = payment value = "P2">
          Master Card
     <input type = "radio" name = payment value = "P3">
          무통장 입금
   </blockquote>
   <hr>
   <input type = "submit" value = "주문">
   <input type = "eset" value = "다시 작성">
   </form>
   </body>
</html>
```

(그림 10) 온라인 서점 HTML 소스 코드

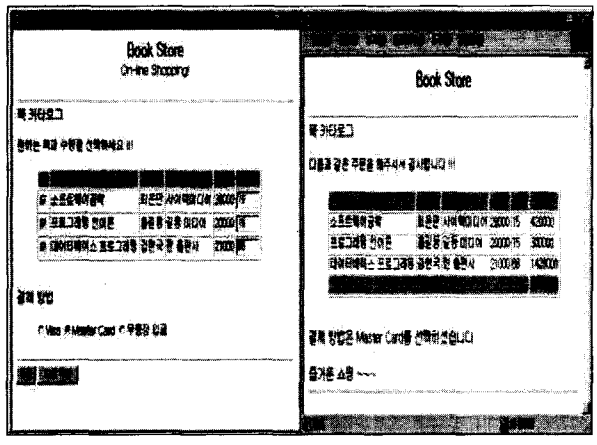
(그림 13)은 (그림 10)에 대한 테스트 스크립트이다. 실제 테스팅은 입력 폼의 각각의 파라미터(수량, 결제 방법 등)에 임의의 난수를 발생시켜 주문 처리를 시뮬레이션 한다. 그리고 이것은 온라인 서점 어플리케이션에 넘겨져서 처리 결과를 기다리게 된다. 테스팅 횟수를 지정해서 여러 번 테스트를 할 수 있다.

(그림 11)은 (그림 10)의 테스트 스크립트를 브라우저에서 실행한 테스트 프로그램이다. (그림 12)는 웹 인터페이스 테스트 프로그램에서 호출하여 실행한 결과 화면이다.

자동화된 테스팅 사례를 알아보았다. 여전히 다른 어플리케이션의 사용자 인터페이스 부분을 위한 테스트 함수를 그에 맞게 다시 만들어야 하는 단점은 있지만 한 번 테스트 스크립트를 만들면 자동으로 여러 번 테스트를 수행할 수 있다는 장점이 있다.



(그림 11) 웹 인터페이스 테스팅 프로그램



(그림 12) 테스트 결과 화면

```
<html>
 <head>
   <title> On-line Book Store Tester </title>
   <script language = "Javascript">
     var Win, RepWin ;
     var count = 0, noOfTests ;
     // ------------------------------------------
     // TEST ENGINE -- Contains Main Function, Tester
Functions and
     // randomly generated Test cases
     // ------------------------------------------
     // Main Function - Test Shop
     //   1. Generate and apply a random order
     //   2. Wait for the response
     //   3. Get back the Shop for next test
    function test( task ) {
      if ( task == 1 ){
         submitAShoppingReq( ) ;
         task = 2 ;
       }
      if ( task == 2 ) {
         var ret = goBack() ;
         if ( ret == false ) {
            setTimeout("test(2)," 2000) ;
            return false;
         }
```

```
        }
        count++ ;
        // Move to next order
        if ( count <= document.forms[0].testCount.value) {
            setTimeout("test(1)", 500) ;
        }
            return true ;
    }
// ---------------------------------------------------
// Tester Function 1 - Open Shop
// ---------------------------------------------------
function init() {
 Win = window.open("http://localhost/shop/bookstore.html",' ',
                    'width = 600, height = 500') ;
}
// ---------------------------------------------------
// Tester Function 2 - Generate a random order and apply to
   the Shop.
// ---------------------------------------------------
   function submitAShoppingReq() {
    var tmp = Math.round(Math.random()×10) ;
    var payment = tmp%3 ;
    Win.document.forms[0].reset() ;
    Win.document.forms[0].bookitem[0].checked = true ; ← ④
    Win.document.forms[0].bookitem[1].checked = true ;
    Win.document.forms[0].bookitem[2].checked = true ;
    Win.document.forms[0].Q1.value =
            Math.round(Math.random()×300) ; ← ⑤
    Win.document.forms[0].Q2.value =
            Math.round(Math.random( )×200) ;
    Win.document.forms[0].Q3.value =
            Math.round(Math.random( )×100) ;
    Win.document.forms[0].payment[payment].checked =
            true ; ← ⑥
    Win.document.forms[0].submit( ) ;
   }
// ---------------------------------------------------
// Tester Function 3 - Close Shop
// ---------------------------------------------------
   function term( ) { Win.close( ) ; }
   function goBack( ) {
      // Check if the response is received
      if ( typeof(Win.opOver) == 'undefined' ) {
          return false ;
      }
      // Check for success of the operation
      if ( Win.document.forms[0].result.value != "Success" ) {
          alert("Operation not Successful") ;
      }
      Win.history.go(-1);
      return true;
   }
</script>

</head>
  <body bgcolor = "#ffffff">
      <center>
      <font size = 5> On-line Book Store </font><br>
      <font size = 4> Web Interface Tester </font><br>
      </center>
   <br><hr>
<form name = Tester onSubmit = "parent.test( )">
    <input type = button value = "Open the Shop!"
          onClick = "init( ) ; "><br>
       No. of Tests
       <input type = text name = testCount size = 5 value = 1>
       <input type = button value = "Test!" onClick = "count = 1 ;
             test(1)" ; ><br>
       <input type = button value = "Close the Shop!"
             onClick = "term( ) ; "><br>
       <br><hr>
   </form>
 </center>
</body>
</html>
```

(그림 13) 테스트 스크립트

〈표 1〉 웹 테스트 도구 비교 평가

도 구	테스트모델	플랫폼	UI테스트 데이터 생성	테스트데이터생성방법
Astra Site Test (LoadRunner, SiteManager)	스트레스테스트, 웹 통합테스트	Windows	불가능	없 음
SilkTest	기능 및 리스레션 테스트	Windows	불가능	없 음
SiteRuler	링크 및 HTML 호환성 체크	Windows, Unix	불가능	없 음
W3C Validator	온라인 HTML 화환성 링크 테스트		불가능	없 음
VisualTest	GUI 성능테스트	Windows, Unix	가능(성능 테스트)	사용자입력
본 연구 개발 시스템	UI Interaction 테스트	Windows, Linux	가능	생 성

5. 구현 시스템의 평가

웹 기반 소프트웨어를 테스트하는 여러 가지 도구가 개발되어 있으나[17] <표 1>에 나타낸것 처럼 테스트 모델이 각기 크게 다르다. 사용자 인터액션을 기반으로 소프트웨어의 기능과 UI를 테스트하는 도구는 VisualTest가 이 논문의 연구와 견줄만하다.

VisualTest의 경우 사용자 인터페이스를 중심으로 테스트한다는 면에서는 이 논문과 같으나 주로 GUI의 성능테스트를 위한 것이며 테스트 데이터도 사용자 입력을 캡춰하여 사용하고 있다. 따라서 이 논문에서와 같이 테스트 케이스가 다양하지 못하다.

웹 기반 시스템의 경우 로그인이 공통적인 기능이므로 테스트가 중요하다. 따라서 랜덤 테스트 케이스보다는 준비된 다양한 자료에 의하여 테스트하는 것이 효과적이다. 연구에서는 테스트 자료를 타입별로 분류하여 데이터베이스로 준비한 후 이를 이용하였다.

6. 결론 및 향후 연구

인터넷의 폭발적인 성장은 웹 어플리케이션의 증가에 영향을 주었다. 웹 관련 기술의 발전과 다양한 비즈니스 요구사항은 웹 어플리케이션을 점점 더 복잡하게 했고, 반면에 웹 어플리케이션의 품질과 신뢰성이 중요해졌다. 따라서 웹 기반 시스템에 대한 효율적인 개발과 유지보수 및 확장을 위해서는 시스템 개발의 생명주기에 따라 적절한 테스팅 방법에 대한 연구가 필수적이다.

본 논문에서는 웹 어플리케이션의 사용자 인터페이스의 테스팅 자동화에 대한 방법을 제시한다. HTML 페이지의 브라우저 내장 객체와 자바 스크립트(JavaScript)의 객체를 매핑(mapping)시켜 사용자 입력 폼에 대한 테스트 케이스

를 자동으로 생성해서 UI 테스팅을 자동화하는 방법을 보여준다. 또한 제안된 방법으로 테스팅 자동화 사례를 보여준다.

효율적인 웹의 사용자 인터페이스 테스팅이 되려면 테스트 스크립트와 테스트 케이스를 생성하는 방법이 필수적이다. 이 논문에서는 자바스크립트를 이용해서 테스트 스크립트를 만들며, 테스트 케이스는 랜덤하게 만들어서 사용하게 된다. 이 기법은 웹 어플리케이션을 개발하는 과정에서 손쉽게 자동화된 방법으로 테스팅을 수행하게 해준다. 여전히 각각의 어플리케이션의 특징(feature)에 대한 테스트 함수를 작성하는 데 따른 비용 부담이 있지만, 한번의 노력으로 자주 릴리즈되는 웹 기반 어플리케이션을 위해 고려할 만한 기법이다.

향후 연구로는 다양한 사용자 인터페이스에 대한 테스트 스크립트를 자동으로 생성하는 생성기에 대한 연구, 사용자 이벤트를 기록하고 재생해서 테스트에 사용하게 할 수 있게 하는 것과 테스트 결과 보고서를 자동 생성하는 연구가 진행되어야 하겠다.

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, Mar., 1999.

[2] Robert L. Glass, "Has Web Development Changed the Meaning of Testing?," StickyMinds.com column, Dec., 2000.

[3] Edward Miller, "WebSite Testing," Software Research Inc., http://www.soft.com/eValid/Technology/White.Papers/website.testing.html, 2000.

[4] Andrea MacIntosh, Wolfgang Strigel, "The Living Creature - Testing Web Applications," QA Labs Inc., Jun., 2000.

[5] Hung Q. Nguyen, *Testing Applications on the Web*, John Wiley & Sons, 2001.

[6] T. A. Powell et. al., *Web Site Engineering : Beyond Web Page Design*, Prentice-Hall, 1998.

[7] M. Cartwright, "Empirical Perspectives on Maintaining Web Systems : A Short Review," IEEE Trans. on Software Engineering, Vol.26-8, pp.786-796, Aug., 2000.

[8] B. M Subraya, S. V. Subrahmanya, "Object driven performance testing of Web applications," Proceedings. First Asia-Pacific Conference on Quality Software, pp.17-26, 2000.

[9] C. Kallepalli, J. Tian, "Usage measurement for statistical web testing and reliability analysis," Software Metrics Symposium, METRICS 2001. Proceedings. Seventh International, pp.148-158, 2001.

[10] D. Kung, Chein-Hung Liu, Pei Hsia, "An Object-Oriented Web Test Model for Testing Web Applications," Proceedings. First Asia-Pacific Conference on Quality Software, pp.111-120, 2000.

[11] Stefan Koch, "VOODOO'S INTRODUCTION TO JAVASCRIPT Version 2.5," http://rummelplatz.uni-mannheim.de/~skoch/js/tutorial.htm, 1996~1998.

[12] Bret Pettichord, "Seven Steps to Test Automation Success," http://www.oi.com/~wazmo/papers/seven_steps.html, Jun., 2001.

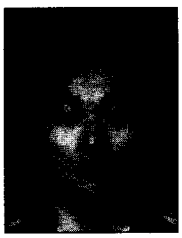
[13] Paul Gerrard, "Risk-Based E-Business Testing : Part 1 Risks and Test Strategy," Systeme Evolutif Ltd., Jun., 2000.

[14] Paul Gerrard, "Risk-Based E-Business Testing : Part 2 Test Techniques and Tools," Systeme Evolutif Ltd., Nov., 2000.

[15] 강재성, 윤광식, 오승욱, 권용래, "웹의 상태기반 기능시험 기법", 정보과학회 봄 학술발표논문집, Vol.27, No.1, pp.501-503, 2000.

[16] 권영호, 최은만, "웹 기반 소프트웨어의 테스트 모델에 관한 연구", 정보처리학회 춘계학술발표논문집, 제8권 제1호, pp. 197-200, 2001.

[17] 최은만, "웹 기반 소프트웨어의 시험 및 검증 기술", 정보과학회지, Vol.19, No.11, pp.19-26, 2001.



권 영 호

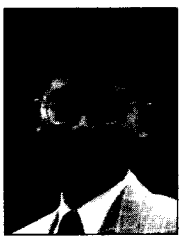
e-mail : yhkwon@dgu.ac.kr

2000년 동국대학교 컴퓨터공학과(학사)

2002년 동국대학교 컴퓨터공학과 (공학석사)

2002년~현재 (주)디티브이인터랙티브 프로세스팀

관심분야 : 프로세스, 테스트 자동화, XP, 데이터 방송



최 은 만

e-mail : emchoi@dgu.ac.kr

1982년 동국대학교 전산학과(학사)

1985년 한국과학기술원 전산학과 (공학석사)

1993년 일리노이 공대 전산학과(공학박사)

1985년~1988년 한국표준연구소 연구원

1988년~1989년 데이콤 주임연구원

2000년~2001년 콜로라도 주립대 전산학과 방문교수

1993년~현재 동국대학교 컴퓨터공학과 부교수

관심분야 : 객체지향 테스트, Program Understanding, 소프트웨어 품질 메트릭, 웹 기반 소프트웨어 테스트