Composition & Transition Rule을 이용한 웹 어플리케이션의 분석 및 테스트 케이스 생성에 관한 연구

김현수° 최은만
동국대학교 컴퓨터멀티미이어공학과
{tetis5°, emchoi}@dgu.edu

A Study on Analysis and Test Case Generation for Web Application by Using Composition & Transition Rule

Hyun Soo Kim°, Eun Man Choi
Dept. of Computer Engineering, Dongguk University

요 약

인터넷을 기반으로 하는 웹 어플리케이션의 급성장으로 웹 어플리케이션의 품질에 대한 요구가 중요시되고 있으며, 웹 기반 어플리케이션에 대한 품질을 보증하는 연구가 활발히 진행되고 있다. 또한 품질을 보증하기 위한 여러 가지 방법이 연구되고 있으며, 테스트를 위한 많은 도구들이 존재하고 있다. 하지만 웹 어플리케이션의 테스트는 웹의 다양한 구성요소라는 특성으로 테스트하기에 어려움이 있다. 이 논문에서는 웹 기반 어플리케이션의 테스팅을 보다 효율적으로 진행하기 위해 웹의 상태를 논리 흐름에 따라 구분하고 Composition & Transition Rule을 적용하여 웹 페이지의 전체적인 테스팅을 커버할 수 있는 테스트 케이스를 생성하는 방법을 제안하고 설명한다.

1. 서 론

오늘날, 인터넷이 정보 검색을 중심으로 전자상거래, 사이버교육 등 다양한 서비스로 활용범위가 넓어지고 있다. 비즈니스 세계에서 인터넷이 점차 중추적인 역할을 하기 시작하면서 E-비즈니스의 성장과 함께 웹 기반 어플리케이션의 품질 보증은 필수적이며 높은 비중을 차지하고 있다. 특히 인터넷을 매개로 한 전자상거래의 경우 웹은 모든 기업의 경쟁력 향상을 위한 필수 조건이 되었다. 웹 어플리케이션은 점점 더 복잡해지고 있으며, 다양한 웹 기반 기술을 가지고 테스팅과 품질 관리를 하기가 점점 더 어려워지고 있는 실정이다.

웹 어플리케이션 테스팅의 중요도에 비해 연구는 부족한 실정이다. 웹 어플리케이션을 테스트하는 방법은 웹 페이지가 정적인지, 동적인지에 따라서 달라질 수 있으며, 정적인 경우는 HTML 구문 검사와 링크 검사로 가능하지만, 동적인 경우에는 통합 테스트가 필요하다[2].

웹 어플리케이션의 테스트 데이터를 생성하는 것은 결코 쉬운 일이 아닌데, 그 이유는 첫째, 웹 어플리케이션을 구성하는 컴포넌트로의 분할이 쉽지 않으며, 둘째, 빈번히 일어나는 변경으로 테스트 노력이 많이 들고 반복적인 작업이 많으며, 셋째, 정적 분석으로 테스트 드라이버의 작성이 가능하나 대부분 네이게이션 링크, 철자법, HTML 호환 오류 등을 찾아내기 위한 목적으로만 사용되고 있고, 넷째, 웹 페이지의 동적인 부분과 정적인 부분을 동시에 테스트하기가 어렵기 때문이다.

본 논문에서는 웹 어플리케이션을 테스트하기 위해서 Composition &Transition Rule과 분기 포인트를 사용하여 웹 어플리케이션의 복잡한 논리 흐름을 최소화하고 이를 통해 웹 페이지 전체의 테스트를 커버할 수 있는 테스트 케이스 생성에 초점을 두고 있다.

2. 관련연구

웹 기반 소프트웨어는 매우 높은 수준의 품질이 요구된다[7]. 웹 어플리케이션은 전통적인 소프트웨어와 많은 차이가 있는데, 웹 어플리케이션은 네트워크 상에서 수행되며 네비게이션과 링크를 제공하며 사용자와 다양한 인터렉션이 이루어지며, 대부분의 웹 테스트는 Javascript validation tools, link checking tools, HTML validators 그리고 capture/playback tools 등을 이용하여 클라이언트 부분의 검증과 정적인 서버 부분의 검증에 초점을 맞추고 있다. 현재 웹 테스팅에 사용하고 있는 기법은 대부분 전통적인 기법-예를 들면, 기능테스트, DB 테스트, 보안 테스트, 성능 테스트-을 사용하고 있고 여러가지 테스팅 도구들이 사용되고 있다[6].

2.1 FSM을 이용한 테스트

웹의 상태 기반 기능 시험은 웹을 기반 모델로 보고, 웹의 동적 행위를 웹의 상태 전이로 정의한다. 이러한 상태 전이를 표현하기 위해 웹이 가질 수 있는 상태 변화 경로를 추출하여 상태 기반 시험 기법을 도입하여 시험 사례를 생성하는 기법이다[3]. 웹 기반 시스템이 사용자가 원하는 기능을 잘 전달하는지를 시험하기에 앞서, 먼저 웹의 상태와 웹 상태 전이 그래프(WSTG)를 정형적으로 정의한다.

W.State=(HTML_page, Web_var)
HTML_Page : 하나의 독립적인 HTML 문서
Web_var : HTML에서 컴퓨팅 매체로 보내는 입력들의 집합

그림 1 웹 사이트 W의 웹 상태

WSTG W=(Ho, H, I, I', T)
H: W.State의 집합
Ho∈H : 초기상태
I : 사용자의 입력 집합
I' : 웹 컴퓨팅 매체로부터 생성되는 입력 집합
T : H × (I, I') → H

그림 2 웹 상태 전이 그래프(WSTG)

WSTG는 동일한 FSM(Finite State Machine)로 변환될 수 있으며 FSM으로부터 테스트 경로를 생성하고 수행이나 상태전이로부터 발생하는 에러를 찾아낼 수 있다.

FSM F=(I, E, Y, T, O)
I : 입력들의 집합
E : 초기상태를 포함하는 상태들의 집합
Y: 출력들의 집합
T : 전이기능, X × E → E
O : out function, X × E → Y

그림 3 Finite State Machine(FSM)

FSM에서 시험 사례를 생성하는 기법을 적용하여 하나씩 웹에서 적용하면서 올바른 상태전이가 일어나는지 검사하면서 시험을 수행한다.

2.2 테스트 스크립트를 이용한 테스트

웹 어플리케이션에서 행위를 테스트 스크립트로 표현하려면 HTML 폼으로 들어오는 데이터 입력과 웹 서버의 응답에 대해 검증하는 두 가지의 테스트 데이터가 필요하다. 웹 어플리케이션에서 입력되는 데이터들을 트랙하는 과정은 다음과 같다[8].

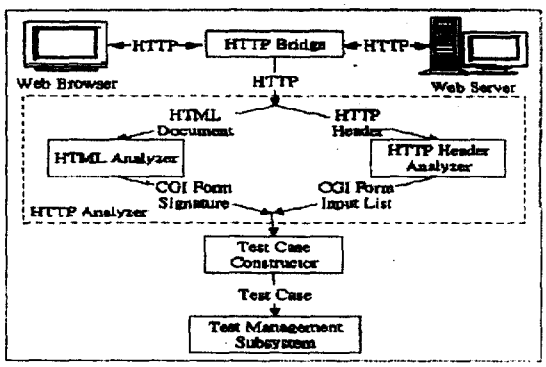

그림 4 데이터 입력 소스 레코더

데이터 입력 소스 레코더는 HTTP Bridge와 HTTP 분석기를 포함하며, HTTP Bridge는 웹 서버와 웹 브라우저 사이의 HTTP를 이용한 정보교환을 캡쳐하고 캡쳐된 정보는 HTTP 분석기로보내져서 분석된다. HTTP 헤더 분석기는 HTML 폼의 입력 값을 추출하기 위해 HTTP에서의 정보교환을 분석하고 HTML 분석기는 HTML 폼 안에서의 필드네임과 같은 폼에 관련된 정보를 추출하기 위해 HTML의 데이터의 입력을 받아들이는 특정한 부분을 분석한다. HTTP 헤더 분석기와 HTML 분석기에서 분석된 결과 값들은 테스트 스크립트를 생성하는데 이용된다.

3. Composition & Transition Rule

웹 어플리케이션 분석 모델은 클라이언트와 서버가 HTML 페이지를 통하여 상호작용을 한다고 보고 웹 페이지를 composite section으로 나누어 동적인 특징과 그들 간의 상호작용을 분석하기 위한 것이다[4]. 상호작용을 분석하기 위해서는 Composition & Transition Rule이 사용된다.

3.1 Composition Rule

Basis : P
Sequence (p→p1 · p2) : p1 에 이어 p2가 온다.
Selection (p→p1 | p2) : p1, p2 둘 중 하나를 선택하며, 동시에 둘을 선택할 수는 없다.
Aggregation (p1(p2)) : p2가 p1에 포함된다.

그림 5 Composition Rule

P^*는 0 또는 더 많은 구역을 조합하는 것을 표현하며, 1 또는 더 많은 구역의 조합은 P^+로 나타낼 수 있다. P^n은 정확히 n번의 조합을 의미하며, $p^{(0|1)}$은 p가 선택적으로 포함될 수 있다는 것을 나타낸다. 예를 들어, P->p1•(p2|p3)*•p4의 의미는 제일 처음 p1이 선택되며, 다음으로 p2 또는 p3가 0번 이상 선택적으로 선택되며, 그 다음으로 p4가 선택된다는 의미이다.

3.2 Transition Rule

Link Transition(p⇒q): 링크를 통한 변이
Composite Transition(s→p): s를 실행시킴으로써 일어나는 변이
Operation Transition(p→q):시스템의 구성으로 야기되는 변이

그림 6 Transition Rule

Transition Rule은 크게 HTML 링크와 사용자의 행위에 의해 이루어지는 변이로 나눌 수가 있다. 위에서 p와 q는 composite section을 말하며, s는 HTML 페이지를 물리적 단위로 나눈 구역을 나타내고 s는 서블릿 또는 다른 소프트웨어의 컴포넌트를 의미한다.

4. 테스트 케이스 생성

atomic section은 정적인 HTML 페이지 또는 HTML로 출력되는 서버 프로그램의 section을 말한다[4]. 하지만 atomic section을 위와 같이 정의하면 웹 페이지의 다양한 논리적인 흐름을 정확히 파악할 수 없으므로 논리 흐름에 따라 atomic section을 정의하도록 하겠다.

SearchEngines 서블릿은 reportProblem(),doGet(), doPost(),SearchSpec()의 메서드를 가지고 있으며, SearchEngines 서블릿을 Composition & Transition Rule을 사용하여 테스트 케이스를 생성하면 논리 흐름에 따라 많은 테스트 케이스가 생성될 수 있다. 예를 들어, 정상적인 경우, Search String이 없는 경우, Search String이 있는 경우, numResults가 있는 경우, 없는 경

표 1 SearchEngines 서블릿의 atomic section

P1	public class SearchEngines extends HttpServlet {
	 public void doGet(HttpServletRequest request, HttpServletResponse response)
	 throws ServletException, IOException {
	 String searchString = request.getParameter("searchString");
	 if ((searchString == null) || (searchString.length() == 0)) {

P2	reportProblem(response, "Missing search string.");
	 return;
	 }

P3	searchString = URLEncoder.encode(searchString);
	 String numResults = request.getParameter("numResults");
	 if ((numResults == null) ||
	 (numResults.equals("0")) ||
	 (numResults.length() == 0)) {

P4	numResults = "10"; }

P5	String searchEngine =
	 request.getParameter("searchEngine");
	 if (searchEngine == null) {

P6	reportProblem(response, "Missing search engine name."); return; }
P7	SearchSpec[] commonSpecs = SearchSpec.getCommonSpecs(); for(int i=0; i<commonSpecs.length; i++) {
P8	SearchSpec searchSpec = commonSpecs[i]; if (searchSpec.getName().equals(searchEngine)) {

P9	String url = searchSpec.makeURL(searchString, numResults);
	 response.sendRedirect(url);
	 return;
	 }
	 }

P10	reportProblem(response, "Unrecognized search engine."); }

우 그리고 이들을 모두 조합한 경우 등 논리 흐름에 따라 복잡한 테스트 케이스가 생성될 수 있다. 이러한 테스트의 복잡성을 줄이기 위해 본 논문에서는 논리 흐름에서의 분기 포인트를 이용하여 보다 효율적이고 간단한 테스트케이스를 생성하고자 한다.

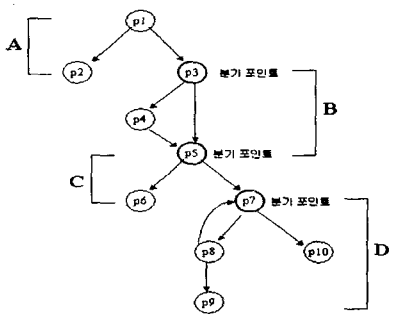

그림 7 SearchEngines 서블릿의 논리 흐름도

즉, 논리 흐름상 다음으로 넘어가기 위해선 반드시 분기 포인트인 p3, p5, p7을 지나쳐야 하므로 분기 포인트를 기점으로 크게 네 구간 A,B,C,D로 Searchengines 서블릿을 나눌 수가 있다. 따라서 A,B,C,D 각각의 테스트 케이스를 생성하면 SearchEnginers 전체의 테스트를 커버하는 테스트 케이스를 생성할 수 있다. Composition & Composition Rule과 분기포인트를 이용하여 테스트 케이스를 생성한 결과는 다음과 같다.

표 2 테스트 케이스

구간	테스트 케이스	내 용
A	p1•p2=>SendError	Search String이 없는 경우
	p1•p3	Search String이 있는 경우
B	p3•p4•p5	NumResult가 없는 경우
	p3•p5	NumResult가 있는 경우
C	p5•p6=>SendError	Search Engine이 없는 경우
	p5•p7	Search Engine이 있는 경우
D	(p7•p8)*• p9=>Result	Search Engine을 올바르게 인식한 경우
	p7•p10=>SendError	Search Engine을 올바르게 인식하지 못한 경우

5. 결론 및 향후 연구

웹 어플리케이션을 테스트하기 위한 웹 어플리케이션을 분석하는 방법과 분기포인트 그리고 Composition & Transition Rule을 사용한 테스트 케이스 생성에 대해 제안하였다. 복잡한 논리 흐름을 포함하고 있는 페이지를 분기포인트를 사용하여 최소한으로 분할하여 복잡한 테스트 케이스를 간략화 할 수 있다. 이로써 보다 효율적인 테스팅을 할 수 있으며 테스팅에 필요한 시간과 노력이 줄어들 것이다.

향후 연구로는 다양한 컴포넌트를 포함한 웹 페이지 테스트에서 분기 포인트와 Composition & Transition Rule을 이용하여 테스트 케이스 자동생성에 관한 연구와 테스트 결과 보고서를 자동으로 생성할 수 있는 연구가 진행되어야 하겠다.

6. 참고 문헌

[1] Jesper Ryden, Par Svensson, "Web Application Testing", Department of Transportation and Logistics Chalmers, 2001.

[2] 권영호, 최은만, "웹 기반 소프트웨어의 테스트 모델에 관한 연구", *정보처리학회 춘계 학술발표논문집*, 제 8권 제 1 호, pp.197-200, 2001.

[3] 강제성, 윤광식, 오승욱, 권용래, "웹의 상태 기반 기능 시험 기법", *한국정보과학회 봄 학술발표논문집*, Vol 1. 27. No. 1, 2000.

[4] Ye Wu, Jeff Offutt, "Modeling and Testing Web-based Applications", 2002, http://www.ise.gmu.edu/~ofut/rsrch/abstracts/web.html.

[5] Filippo Ricca, Paolo Tonella, "Analysis and Testing of Web Applications", *Proceedings of the 23rd International Conference on Software Engineering*, 2001.

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

[7] Jeff Offutt, "Quality attributes of web software applications", *IEEE Software*: Special Issue on Software Engineering of Internet Software, 19(2):25-32, March/April, 2002.

[8] Ji-Tzay Yang, Jiun-Long Huang, Feng-Jian Wang, William C.Chu "Construction an Object-Oriented Architecture for Web Application Testing", *Journal of Information Science and Engineering*, Vol.18 No.1, pp. 59-84, 2000.