정형적 명세를 이용한 웹 프로그램의 테스트

안 영 희, 최 은 만
{yhahn, emchoi}@dgu.ac.kr

Testing Web Program Using Formal Specification

Young-Hee Ahn, Eun Man Choi
Dept. of Computer Multimedia Engineering, Dongguk University

요 약

정형적 명세를 이용하면 원시코드의 복잡함에 방해 받지 않고 필요한 구현 정보를 테스트 프로그래머가 얻을 수 있다. 특히 웹 기반 소프트웨어는 정형적 명세로 시스템에 대한 외부 입력과 반응을 잘 나타낼 수가 있다. 복잡하고 구성요소가 다양한 웹 프로그램의 기능을 정형적 명세를 이용하여 핵심적으로 나타내고 이를 이용하여 웹 프로그램의 실행 동작을 테스트할 수 있는 과정을 제안하고 실험하였다. 실험 대상은 웹 뱅킹 업무로 정하고 정형화 명세에서 상태 천이도를 구성하고 테스트 시나리오를 추출하는 방법을 기술하였다. 제안한 방법은 웹 프로그램의 사용 기반 테스트 기법과 결합하여 웹 소프트웨어의 테스트 자동화에 중요한 요소가 될 수 있다.

1. 서 론

여러 가지 다양한 자료의 접근이 쉬운 인터넷 환경에서 모든 소프트웨어를 사용한다는 것은 하나의 환경과 인터페이스로 통합한다는 의미에서 매력적이다. 또한 클라이언트나 서버에 극단적으로 부담을 주지 않고 적절히 컴퓨팅을 분산시키며 어느 곳에서나 접근하여 사용할 수 있다는 면에서 웹 기반 소프트웨어는 장점이 많다. 이러한 이유로 인터넷뿐만 아니라 인트라넷 환경에 웹 기반 소프트웨어들이 많이 개발되고 있다.

이러한 웹 기반 소프트웨어는 다음과 같은 몇 가지 이유로 화이트 박스 형태의 완벽한 테스트가 현실적으로 어렵다. 첫째로 웹 기반 소프트웨어는 구성 요소가 다양하다. 간단한 HTML로부터 Java Applet, Javascript, CGI, ASP에 이르기까지 다양한 컴포넌트들이 분산 존재하여 그의 추적 및 독립적인 단위 테스트와 분산된 컴포넌트의 통합 시험이 복잡하게 된다[1]. 둘째는 마크업 언어와 스크립트 언어는 절차적 프로그램과는 달리 실행 경로를 파악하기가 쉽지 않다는 문제가 있다. 즉 스크립트 타입의 언어에서 함수의 정의가 태그 안에서 사용될 때 어떤 순서로 사용될 것인지 파악하기가 어렵다.
정형적 명세를 이용하면 원시코드의 복잡함에 방해 받지 않고 필요한 구현 정보를 테스트 프로그래머가 얻을 수 있다. 웹 기반 소프트웨어를 이루는 요소들의 통합을 위한 시험이 끝난 후 기능 시험을 위한 시스템 테스트 단계에는 복잡한 원시코드를 볼 여유가 없다. 특히 웹 기반 소프트웨어는 복잡한 구성요소를 자세히 검토하는데 많은 노력과 시간이 소요되므로 시스템에 대한 외부 입력과 반응을 간결하게 나타내는 방법이 필요하다. 이것이 바로 정형적 명세이다.

웹 기반 프로그램을 상태 천이도로 나타내서 이를 바탕으로 테스트하려는 연구[2][3]는 다음과 같은 문제를 안고 있다.

- 웹 응용의 규모가 커지면 상태 천이도로 나타낼 때 상태가 매우 많아지고 이에 따라 테스트 경로를 찾기가 쉽지 않다.
- 상태천이도(State Transition Diagram)에만 의존하여 테스트 시나리오를 찾아내는 경우 각 기능이 모두 테스트 되었는지 확인하기가 어렵다. 또한 상태 천이를 유도하는 입력이나 외부 자극을 찾아내기는 쉽지만 기능 수행 후 예상되는 결과는 상태천이도만으로 쉽게 찾아 낼 수가 없다.

이 연구에서는 정형적 명세를 웹 소프트웨어의 테스트 과정에 도입하여 이와 같은 문제점을 해결하였다. 정형적 명세를 이용하여 테스트하려는 연구는 일반적인 모형 기반 테스트 프레임 워크를 제안한 연구[4]와 실시간 제약사항과 Z 명세 언어를 결합하여 시스템을 테스트하는 과정을 제안한 연구[5], 객체지향 프로그램의 테스트에 명세를 도입한 연구[6]가 있다. 하지만 웹 기반 소프트웨어에 적용하려는 노력은 없었다.

Object-Z 정형적 언어를 동원하여 웹 프로그램의 명세를 표현하고 여기서 상태 다이어그램을 추출한 후 테스트 시나리오를 작성하여 테스트하는 과정을 고안하였다. 실험 사례로 웹 뱅킹을 시험하였고 그 결과를 나타내었다.

2. 정형적 명세를 이용한 테스트

일반적으로 정형적 명세란 시스템의 속성이나 기능을

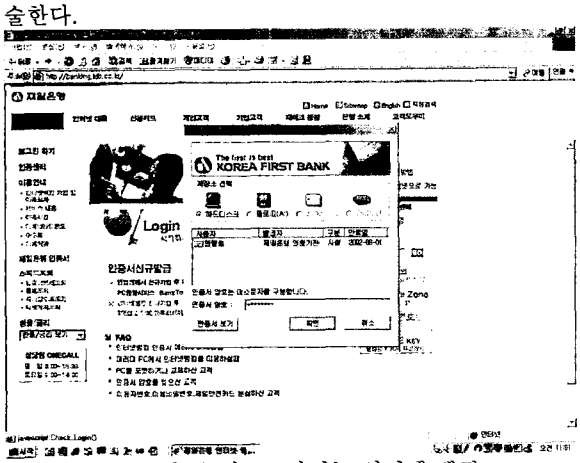
추상화하여 표현한 것으로 추상화 수준이나 개발 단계의 시스템에 대한 관심 대상에 따라 달라진다. 예를 들어 시간이 흐르면서 변하는 시스템의 동작을 나타내기 위하여 temporal logic을 동원하는 히스토리 기반 명세[ ]가 있다. Z, VDM, B 등과 같이 어떤 순간에 시스템이 가져야 하는 변하지 않는 속성(invariant)과 전제조건(pre-condition) 및 종료조건(post-condition)을 나타내는 방법은 상태기반 방법이다. 또한 유한 상태 기계(finite state machine)로 표현하여 입력 이벤트에 대한 상태의 변환을 표현하는 변환 중심 명세(transition-based specification)와 논리 함수로 표현하는 기능 명세 방법이 있다.

네 가지 표현 방법 중 테스트에 잘 적용될 수 있는 것은 상태기반 방법과 변환 중심 명세 방법이다. 변환 중심 명세 방법은 테스트 경로가 무한으로 많아지며 상태를 줄이는 것이 쉽지 않다. 이에 비하여 Object-Z와 같은 상태기반 명세 방법은 구현된 소프트웨어가 가져야 할 기능적인 정보를 잘 나타내므로 블랙 박스의 기능 테스트 오러클을 찾아내기가 쉽다. 그 방법은 다음과 같은 단계로 나뉜다.

1. 웹 기반 소프트웨어의 중요 기능을 Object-Z로 표현한다.
2. 웹의 각 페이지와 매칭되는 STD를 작성한다.
3. STD에서 테스트 시나리오를 추출한다.
4. 테스트 시나리오에 맞는 입력 및 예상 출력 값을 정한다.

3. 웹 프로그램의 명세화

정형적 명세 기법이 웹 프로그램을 테스트하는데 어떻게 이용되는지 그림 1과 같은 인터넷 뱅킹을 예로 들어 설명한다. 인터넷 뱅킹 업무의 예는 소프트웨어 시스템의 기능성에 대한 필요조건을 Object-Z 명세를 사용하여 기술한다.



(그림 1) 테스트하려는 인터넷 뱅킹

예로 든 인터넷 뱅킹의 요구 기능은 다음과 같다.

1. 공인된 보안성:
   - 인증서 암호입력/확인
   - 계좌이체 암호입력/확인
   - 비밀번호 세 번 이상 오류 시 서비스 정지
2. 거래 기능 :
   - 잔액조회
   - 계좌이체

명세는 고객과의 커뮤니케이션을 통하여 정해진다. 최초의 기술과 객체지향분석을 기반으로 한 Object-Z 정형적 명세는 오브젝트와 오퍼레이션을 정의하는데 사용된다. 그러나 웹 기반 소프트웨어에서는 객체지향 시스템의 경계 객체(boundary object)에 해당되는 외부 UI(예를 들면 인증을 위한 PinVerify, 계좌번호를 입력하는 GetAccount)에 관련된 객체를 Object-Z로 그림 2와 같이 나타낼 수 있다.

WEBProcess

$bank : Bank$
$certificateReader : CertificateReader$
$fraudUsed : BankcertificateNo \mapsto B$
$State ::= idle \mid certvlided \mid certpwdverified \mid receivedInfo$

$saveAcc, checkAcc, creditAcc, nullAcc : Account$
$badTry : N$
$state : State$

INIT

$state = idle$
$badTry = 0$
$saveAcc = checkAcc = creditAcc = nullAcc$

PinVerify

$\Delta(state)$
$InPin? : PinType$

$state = securitycardMdied$
$((badTry = 3 \wedge certificateReader.HoldCertificate \wedge badTry' = 0 \wedge state' = idle \vee$
$((badTry < 3) \wedge (inPin? = bank.GetPin \wedge badTry = 0 \wedge state' = pinreceived) \vee$
$(inPin? + bank.GetPin \wedge badTry = badTry + 1 \wedge state' = state))$

GetAccount

$\Delta(state, saveAcc, checkAcc, creditAcc)$

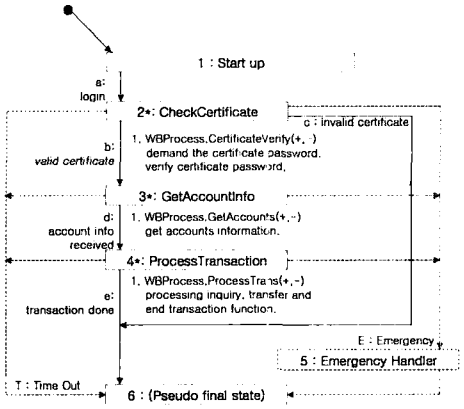
$state = pinverified$
$saveAcc' = bank.GetSaveAcc \parallel bank.GetAccInfo$
$checkAcc' = bank.GetCheckAcc \parallel bank.GetAccInfo$
$creditAcc' = bank.GetCreditAcc \parallel bank.GetAccInfo$
$state' = receivedinfo$

(그림 2) 웹 뱅킹의 정형적 표현

4. 명세 이용 테스트

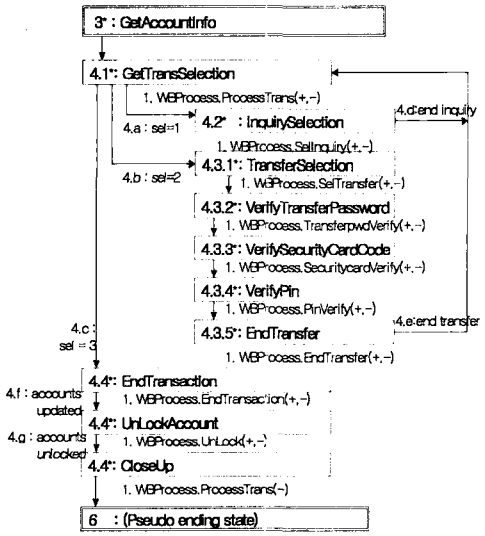
다음은 Object-Z 명세로 표현된 웹 뱅킹의 각 프로세스를 분석하여 STD를 작성하여야 한다. 웹 뱅킹으 프로세스는 기능을 잘 분석하고 이를 위한 웹 페이지에 잘 대칭시켜야 한다. 예를 들면 구좌번호를 선택하거나 입력하는 화면은 그림 2의 GetAccount 프로세스로 정형화할 수 있고 STD에서는 그림 3에서 상태 4와 같이 GetAccountInfo 상태로 나타낼 수 있다.상태를 변화시키는 트랜지션은 대부분 IBProcess에 포함된 프로세스, 예를 들면 GetAccount, ProcessTransaction(웹 뱅킹의 경우 계좌이체, 잔액 조회 등)과 같은 것이다.

상태를 천이시키는 트랜지션에 표시된 +는 프로세스에 수행에 들어갔다는 것을 의미하며 - 는 퇴장을 의미한다. 자세한 변환과정은 [7]에 잘 나와 있다.



(그림 3) STD: 웹 뱅킹의 최상위 레벨

그림 3에 표현된 최상의 레벨에서 웹 뱅킹의 중요한 핵심 기능은 상태 5이다. 상태 5를 더 상세히 분석하면 그림 4와 같이 된다. 계좌이체 또는 잔액 조회를 위한 트랜젝션 선택이 5.1이며 선택 후 적절한 상태로 들어간다. 트랜젝션이 끝난 뒤에는 계좌를 닫고 로그아웃한다. STD는 하향식으로 계속 상세화하여 추가할 수 있다. 예를 들면 5.4의 계좌이체에 대한 과정도 5.4.1 이체 번호 및 금액 입력, 5.4.2 이체 구좌 및 한도 확인, 5.4.3 실제 이체 등으로 나눌 수 있다.



(그림 4) 상태 4에 대한 상세 STD

(표 1) 웹 뱅킹 테스트 시나리오의 예

Test Scenario

$v_1 v_2 v_3 v_4 v_{5.1.1} v_{5.1.2} v_{5.1.3} v_{5.2.1.1} v_{5.2.1.2} v_{5.2.2} v_{5.2.3} v_{5.2.4.1} v_{5.2.4.2} v_{5.2.4.3} v_{5.2.4.3'} v_{5.2.4.4} v_{5.2.4.5} v_{5.2.4.6} v_{5.2.9} v'$

Current State	function list	Next State	Event
1	{ }	2*	open login page(button click)
2*	ib InfoVerify(+,-)	3*	valid ifo
3*	ib PinVerify(+,-)	4*	valid PIN entered
4*	ib GetAccount(+,-)	5.1.1	account information received
5.1.1	ib ProcessTrans(+)	5.1.2	starting transaction selection
5.1.2	messageScreen.DisplayBloc	5.1.3	transaction menu displayed
5.1.3	k(+,-)	5.2.1.1	*acc transfer selection entered*
5.2.1.1	keyInput.GetOption(+,-)	5.2.1.2	starting transfer transaction
5.2.1.2	Transfer(+)	5.2.3	
5.2.2	ib.SetTransNum(+)		
5.2.3	keyInput.GetCertificateNum		certification completed
5.2.4.1	ber	5.2.4.3	transfer account entered
5.2.4.3	saveAcc Transfer(+)	5.2.4.4	valid transfer account
5.2.4.4	ib.SetTransNumber(+)	5.2.4.5	acc transfer started
5.2.4.5	keyInput.GetAccount(+,-)	5.2.4.6	acc transfer
5.2.4.6	accTransfer TransferAcc(+)	5.2.9	
5.2.9	accTransfer TransferAcc(-)	5.1.2	
5.1.2	saveAcc Transfer(-)	5.1.3	
5.1.3	ib Transfer(-)	5.6.1	
5.6.1	messageScreen.DisplayBloc	5.6.2	
5.6.2	k(+,-)	5.6.3	
5.6.3	keyInput.GetOption(+,-)	5.7.1	
5.7.1	ib EndTransaction(+,-)	5.8	
5.8	ib UpdateSet(+,-)	5.9	
5.9		7	
7			

표 1은 그림에서 추출한 계좌이체를 위한 테스트 시나리오이다. 상위레벨의 STD에서부터 시작하여 상태의 변화를 따라 천이를 위한 기능리스트와 입출력 이벤트를 찾아낸다. 사용자 선택과 입력값에 따라 다음 상태를 찾아내고 다시 작업을 반복하면 테스트 시나리오를 얻을 수 있다.

5. 결론과 향후 연구

웹 사이트가 정형적으로 표현 가능하여 여기서 테스트 시나리오를 얻을 수 있음을 보였다. 웹 사이트는 불특정 다수의 사용자에 의하여 사용되며 계속 변경이 일어나므로 사용 기반 테스트 방법과도 병행할 수 있다.

참고 문헌

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

[2] 권영호, 최은만, "웹기반 소프트웨어의 테스트 모델에 대한 연구", 2001 춘계학술대회 논문집, 한국정보처리학회, pp.

[3] H. S. Hong, Y. R. Kwon, and S. D. Cha. *Testing of objectoriented programs based on finite state machines.* In Proceedings of the Second Asia-Pacific Software Engineering Conference (Brisbane, Australia, December 6--9), pages 234--241, 1995.

[4] P.A. Stocks, D. Carrington, Test templates: A specification-based testing framework, Proceedings of the 15th International Conference on Software engineering, 1993, pp.405-414.

[5] D. J. Richardson, S.L. Ahs, T.O.O'Malley, "Specification-based test oracles for reactive system", Proc. Of 14th ICSE, 1992, pp.1-5-118.

[6] K. Chang, et. Al, "Testing object-oriented programs: from formal specification to test scenario generation", Journal of Systems and Software, 42, 1998, pp.141-151.

[7] C. Chen, Derivation of State Transision Diagrams from Object-Z Specifications, Master's thesis, Auburn Univ. 1996.