제 18 회 한국정보처리학회 추계학술발표대회 논문집 제 9 권 제 2 호 (2002.11)

정형적 명세를 이용한 웹 프로그램의 테스트

안영희, 최은만
동국대학교 컴퓨터공학과
e-mail : {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

이 논문에서는 정형적 명세를 이용하여 테스트 데이터를 추출하는 방법을 제안한다. 복잡하고 구성요소가 다양한 웹 프로그램의 기능을 Object-Z 정형 명세 언어를 이용하여 핵심적으로 나타낸다. 이로부터 상태 모델을 구성하고 최상위 레벨의 STD 에서 세부적으로 STD 를 추가하여 테스트 시나리오를 추출한다. 실험 대상은 웹 뱅킹 업무로 정하고 계좌개설 과정의 테스트 데이터를 추출하였다. 제안한 방법은 사용기반 테스트 기법과 결합하여 웹 소프트웨어의 테스트 자동화에 중요한 요소가 될 것이다.

1. 서론

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 HTML 로부터 Java Applet, Javascript, CGI, ASP 에 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 방해받지 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 그 규모가 커지면 상태가 매우 많아지고 테스트 경로를 찾기가 어렵다는 문제점이 있다. 또한 상태천이를 유도하는 입력이나 외부 자극을 찾아내기는 쉽지만 수행 후 예상되는 결과는 상태천이도만으로 쉽게 찾아낼 수가 없다.

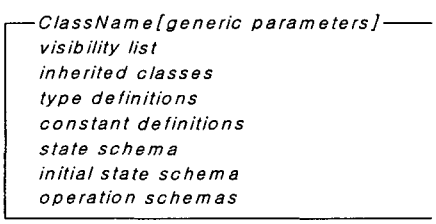
이 연구에서는 정형적 명세를 웹 소프트웨어의 테스트 과정에 도입하여 이와 같은 문제점을 해결하였다. 정형적 명세를 이용하여 테스트하려는 연구는 일반적인 모형 기반 테스트 프레임 워크를 제안한 연구[4]와 실시간 제약 사항과 Z 명세 언어를 결합하여 시스템을 테스트하는 과정을 제안한 연구[5], 객체지향 프로그램의 테스트에 명세를 도입한 연구[6]가 있다. 하지만 웹 기반 소프트웨어에 적용하려는 노력은 없었다.

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 명세

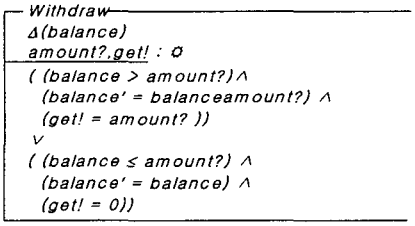
비정형 명세는 시스템의 요구기능을 나타내기 위하여 다이어그램이나 텍스트, 테이블, 기호 등을 사용하는 반면 정형적 명세는 수학적 형태의 신택스와 시맨틱스로 오퍼레이션과 시스템 행위를 표현한다. 이러한 수학적 표기법의 사용으로 정형적 명세는 더욱 정확하고 세밀한 기능 표현이 가능하며 비수치적인 제약조건까지도 표기가 가능한 것이다.

정형적 명세 언어에는 Z 나 VDM 과 같은 모델 지향 정형 명세 언어와 ANNA 나 temporal logic 과 같은 프로퍼티 지향 정형 명세 언어가 있다. 이 논문에서는 Object-Z 를 채택하였는데 이는 모델 지향 명세 언어로 잘 알려진 Z 를 기반으로 하는 객체지향 명세 언어이기 때문이며 객체지향 개념이 잘 표현되고 웹 상에서 실제 사례와 연구 자료를 쉽게 얻을 수 있기 때문이다. 시스템의 Object-Z 명세는 박스 형태의 클래스 스키마로 구성되며 그 구조를 그림 1 에 나타내었다.



(그림 1) Object-Z 클래스 스키마 구조

다른 정형적 명세 언어와 마찬가지로 Object-Z 는 구현정보가 부족하기 때문에 화이트박스 테스팅에는 부적합하며 기능적인 정보를 잘 나타내므로 블랙박스 테스팅에 유용하게 이용될 수 있다. 그림 2 에 Object-Z 오퍼레이션 스키마를 나타내었다. 오퍼레이션의 이름은 *Withdraw* 이다.



(그림 2) Object-Z 오퍼레이션 스키마

Δ기호는 스키마에 의해 변경되는 오브젝트의 데이터 속성을 의미한다. *Withdraw* 오퍼레이션은 입력값으로 amount?을 받아서 get!을 출력하는데 '?'는 amount?가 스키마의 입력 파라미터, '!'는 get!이 출력 파라미터라는 것을 의미한다. '''은 프라임이 붙지 않은 같은 변수의 다음 상태를 나타낸다. 스키마는 전제조건과 종료조건으로 표현되며 그림 2 에서 (balance>amount?)는 전제조건이고 (balance'=balanceamount?)와 (get!= amount?)는 종료조건이다. Object-Z 명세는 오퍼레이션의 전제조건과 종료조건 뿐만 아니라 입출력도 표현 가능함을 알 수 있다.

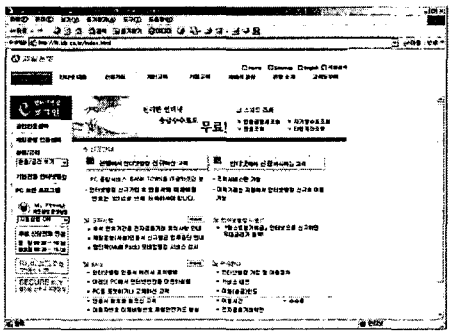
Object-Z 명세는 구현된 소프트웨어가 가져야 할 기능적인 정보를 잘 나타내므로 블랙 박스의 기능 테스트 오러클을 찾아내기가 쉽다. 본 논문에서 제안하는 방법은 다음과 같은 단계로 나뉜다.

1. 시스템의 중요 기능을 Object-Z 로 표현한다.
2. 웹의 각 페이지와 매칭되는 STD 를 작성한다.
3. STD 에서 테스트 시나리오를 추출한다.
4. 입출력 자료사전을 작성한다.
5. 테스트 데이터를 추출한다.

3. 웹 프로그램의 명세화

웹 프로그램을 정형적으로 표현하여 테스트가 가능한지 설명하기 위하여 그림 3 의 인터넷 뱅킹 업무를 예로 들겠다. 사용하는 인터넷 뱅킹의 요구 명세는 다음과 같다.

- 이용자는 유효한 인증서가 있어야 한다.
- 인증서암호, 이체비밀번호, 통장비밀번호, 안전카드코드 입력/확인
- 인증서 암호 5 회 연속 오류 시 재시도
- 계좌유형 : 입출금식, 당좌, 정기, 적립식 예금
- 서비스 종류 : 조회, 이체, 신용카드, 예금신규



(그림 3) 테스트하려는 인터넷 뱅킹

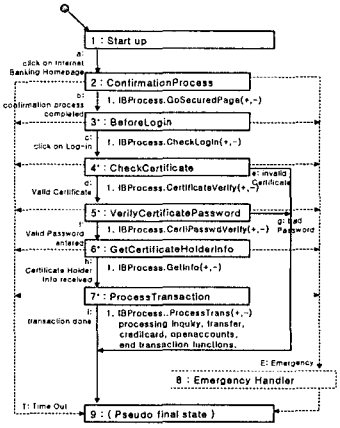
인터넷 뱅킹에서의 물리적 객체는 Object-Z 클래스로 표현되며, *WebBrowser, Keyboard, Mouse, Account, Bank, IBProcess* 등이 이에 해당된다.. *IBProcess* 의 속성은 bank, account, 효력 정지된 인증서를 나타내는 partial function 을 포함한다. *IBProcess* 는 크게 네 가지 상태로 나타낼 수 있으며 상태간의 천이는 외부 이벤트와 내부 오퍼레이션에 의해서 일어난다. 로그인하여 인증서를 받기 전까지는 idle 상태이다. 이것은 인증서가 유효한지 체크하는 상태로 천이되고 인증서가 유효하면 *IBProcess* 는 *certivldted* 상태로 된다. 여기서 인증서 암호 입력을 요구하는데 인증서 암호가 맞으면 내부 오퍼레이션에 의해 계좌정보를 받게 된다.

Account 객체는 상태변수들로 기술되며 *badTry* 는 잘못된 PIN 입력회수를 의미하고 state 는 *IBProcess* 의 상태를 나타낸다. *Inquiry* 의 경우는 계좌 하나만 선택하면 되지만 *Transfer* 의 경우는 두 계좌를 선택하여 출금계좌에서 *Withdraw* 하고 입금계좌에 *TransferDeposit* 한다. 웹 뱅킹의 Object-Z 명세를 그림 4 에 나타내었다.

(그림 4) 인터넷 뱅킹의 Object-Z 명세

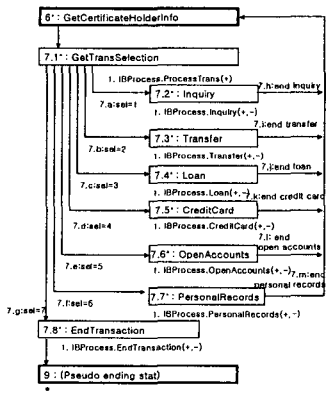
4. 명세 이용 테스트

IBProcess 명세와 사용 정보를 기반으로 최상위 레벨 STD 를 작성하면 그림 5 와 같다. 상태를 변화시키는 트랜지션은 대부분 IBProcess 에 포함된 프로세스, GetAccount, ProcessTransaction(웹 뱅킹의 경우 조회, 이체, 신용카드, 예금신규 등)과 같은 것이다. 상태를 천이시키는 트랜지션에 표시된 +는 프로세스가 수행에 들어갔다는 것을 의미하며 -는 퇴장을 의미한다. 그림 3 에 표현된 최상위 레벨에서 웹 뱅킹의 중요한 핵심 기능은 상태 7 이다.



(그림 5) STD: 웹 뱅킹의 최상위 레벨

상태 7 을 더 상세히 분석하면 그림 6 과 같이 된다. 조회 또는 이체, 예금신규 등을 위한 트랜잭션 선택이 7.1 이며 선택 후 적절한 상태로 들어간다. 적립식 예금의 신규과정을 예로 들면 7.6.1 예금신규, 7.6.2 적립식 예금, 7.6.2.1 신규계좌 입금금액, 7.6.2.2 신규계좌 비밀번호, 7.6.2.3 만기일자, 7.6.2.4 출금계좌번호 및 출금계좌비밀번호 입력, 7.6.2.5 이체비밀번호 입력, 7.6.2.6 안전카드코드 입력 등으로 STD 는 하향식으로 계속 상세화하여 추가할 수 있다.



(그림 6) 상태 7 에 대한 상세 STD

Object-Z 명세로부터 STD 를 작성한 다음 테스트 시나리오를 추출해야 한다. 표 1 이 예금신규를 위한 테스트 시나리오의 예이다.

(표 1) 테스트 시나리오의 예 (예금 신규)

Test Scenario :
$v_1v_2v_3v_4v_5v_6v_{7.1.1}v_{7.1.2.1}v_{7.1.2.2}v_{7.1.2.3}v_{7.1.2.4}v_{7.1.2.5}v_{7.1.2.6}v_{7.1.2.7}v_{7.1.2.8}v_{7.1.3}v_{7.1.4}v_{7.1.5}v_{7.1.6}v_9$

Current State	function list	Next State	Event
1	{}	2*	click on Internet banking home
2*	ib.SecuredPage(+,-)	3*	confirmation process completed
3*	ib.CheckLogin(+,-)	4*	click on Log-in
4*	ib.CheckCertificate(+,-)	5*	*valid* password entered
5*	ib.CertiPasswdVerify(+,-)	6*	certification holder info received
6*	ib.InfoVerify(+,-)	7.1.1	transaction menu displayed
7.1.1	mouse.GetOption(+,-)	7.1.2.1	open account selected
7.1.2.1	ib.SelAccountType(+,-)	7.1.2.2	installment deposit account type selected
7.1.2.2	keyboard.GetAmount(+,-)	7.1.2.3	amount entered
7.1.2.3	keyboard.GetPassword(+)	7.1.2.4	new password entered
7.1.2.4	ib.SelAccount(-)	7.1.2.5	transfer account entered
7.1.2.5	ib.VerifyPin(+)	7.1.2.6	PIN verified
7.1.2.6	ib.VerifyTransferPassword(+)	7.1.2.7	transfer password verified
7.1.2.7	ib.VerifySecurityCardCode(+)	7.1.2.8	security card code verified
7.1.2.8	saveAcc.Transfer(-)	7.1.2.9	transaction completed
7.1.3	ib.EndTranssection(+,-)	7.1.4	closing transaction
7.1.4	ib.UpdateSet(+,-)	7.1.5	accounts updated
7.1.5	bank.EndTranscation(+,-)	7.1.6	bank transaction completed
7.1	ib.ProcessTrans(-)	9	transaction done
9	{}	-	

상위레벨의 STD 에서부터 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. 매칭되는 테스트 시나리오의 입출력 시퀀스는 다음과 같다.

- Test scenario 1 : (id?(SAcc), amount!; nil),
- Test scenario 2 : (id?(FAcc), amount!; amount!),
- Test scenario 3 : (id?(IAcc), amount!, id?(IAcc), amount! ; amount!)

표 2 에 입출력 자료사전의 일부를 나타내었다.

(표 2) 입출력 자료사전

ID	Name	Type	Constraints	Asso. Type
			IBProcess	
SelectAccount				
30	*id?*	?	*id?∈dom savingAccounts*	*SAcc*
31	*id?*	?	*id?∈dom fixedDepositAccounts*	*FAcc*
32	*id?*	?	*id?∈dom installmentDepositAccounts*	*IAcc*
GetAmount				
40	*amount!*	○	>0	NA
OpenAccount				
41	*id?*	?	*1≤ id? ≤ 10000000 id?≠dom savingAccounts*	*SAcc*
42	*id?*	?	*10000001≤ id? ≤ 20000000 id?≠dom fixedDepositAccounts*	*FAcc*
43	*Id?*	?	*20000001≤ id? ≤ 30000000 id?≠dom installmentAccounts*	*IAcc*
44	*amount!*	○	*>0*	NA

5. 실 험

테스트 데이터 생성 방법은 대상 시스템과 입력값의 특성에 근거하여 선택한다. 예의 예금신규 과정을 분석해 보면 입력값은 N 타입의 id(account numbers)? 와 P 타입의 amount? 두 종류라는 것을 알 수 있다. 앞 절에서 예로 든 테스트 시나리오의 테스트 데이터는 다음과 같다.

- Test scenario 1 : (10000001,200000), (2000000, 300000), (10100125, 3000000), (10000235, 1000000)
- Test scenario 2 : (20000001, 500000), (30000000, 20000000), (22034734, 1000000), (23412356, 200000)
- Test scenario 3 : (00000001, 100000, 00001234, 200000), (10000000, 300000, 00034567, 30000), (01001378, 3000000, 02001279, 500000), (02203456, 2000000, 00021555,10000000)

6. 결론 및 향후 연구

웹 사이트가 정형적으로 표현 가능하여 여기서 테스트 데이터를 얻을 수 있음을 보였다. Object-Z 와 같은 상태기반 명세 방법은 구현된 소프트웨어가 가져야 할 기능적인 정보를 잘 나타내므로 블랙 박스의 기능 테스트 오러클을 찾아내기가 쉽다. 웹 사이트는 불특정 다수의 사용자에 의하여 사용되며 계속 변경이 일어나므로 사용 기반 테스트 방법과도 병행할 수 있다.

참고문헌

[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.

[8] J.D.Musa, "Operational profiles in software reliability engineering". *IEEE Software*, March 1994, pp.14-32.

[9] C. Wohlin and P. Runeson, "Certification of Software components", *IEEE Trans. Software. Eng.* 20 (1994) 494-499.

[10] D. Carrington and P. Stocks, "A tale of two paradigms: formal methods and software testing", in Workshops in Computing Series: Z User Workshop, Springer-Verlag, 1994,pp.51-68.

[11] R. Duke, P. King, G. Rose, and G. Smith, "The Object-Z specification language, Version 1", Technical Report 91-1, Software Verification Research Centre, Department of Computer Science, University of Queensland, May 1991.

[12] J.A. Whittacker, "Markov analysis of software specification", *ACM Trans. Soft. Eng. Methodology* 2 (1993) 93-106.

[13] D. Kung, N. Suchak, J. Gao, P. Hsia, Y. Toyoshima, and C. Chen. "On object state testing". In *Proc. 18th Annual Intenational Computer Software and Application Conference*,pages222-227,Taipei,Taiwan,November 1994.