SOA를 위한 테스트케이스 생성 기법

이승훈*, 강동수*, 송치양**, 백두권*
*고려대학교 컴퓨터전파통신학과
**경북대학교 소프트웨어공학과
e-mail : *{neatnesh, 2008010372, baikdk}@korea.ac.kr, **cysong@knu.ac.kr

A Method of Test Case Generation for Service-Oriented Architecture

SeungHoon Lee*, DongSu Kang*, Chee-Yang Song**, Doo-Kwon Baik*
* Department of Computer and Radio Communications Engineering, Korea University
** Department of Software Engineering, Kyungpook National University

요 약

SOA는 최근 급부상한 기술로 발전된 웹서비스 기술과 접목되면서 특히 실시간 기업에게 각광받고 있다. 이에 따라 SOA를 위한 민첩성과 빠른 적응력이 충족되는 구체적인 기법 연구가 많은 진행 중에 있다. 이는 SOA에서의 테스트 역시 마찬가지이며, 본 논문에서는 민첩성과 적응력을 높일 수 있는 SOA를 위한 테스트케이스 생성 기법을 제안한다. SOA는 기존의 컴포넌트 기반의 개발과는 관점의 차이 때문에 절차나 기법의 접근 방법이 다를 수 있다. 따라서 본 논문에서는 SOA의 서비스를 하부 계층인 컴포넌트로 분할하여 기존의 연구를 활용한 후, 다시 서비스 단위로 합성하여 서비스의 테스트케이스를 생성한다. 그리고 제시한 기법을 인터넷 뱅킹 시스템의 계좌잔액조회 서비스에 적용함으로써 서비스 단위의 테스트케이스 생성 사례를 보인다. 제안 기법을 통해 서비스 단위의 테스트를 체계적으로 할 수 있으며, 빠른 릴리즈를 실현하여 SOA의 민첩성과 적응력을 높일 수 있다.

1. 서론

서비스지향 아키텍처(Service-Oriented Architecture)는 최근 급부상한 기술로 많은 주목을 끌고 있으며, 발전된 웹서비스 기술과 접목되면서 여러 기관에게 많은 이득을 가져다 주고 있다. 특히 SOA 는 경쟁력을 극대화하고, 빠른 적응력과 민첩성을 지닌다는 특성을 가진 실시간 기업(Real Time Enterprise)에게 가장 적합한 정보기술 아키텍처로서 각광받고 있다[1]. 하지만 SOA 는 아직도 많은 표준과 체계적인 절차 정립이 필요하다. 새롭게 제안되고, 확립되는 기법들은 민첩성(Agility)과 빠른 적응력을 갖추어 실제로 적용하는 실시간 기업들의 요구에 부응해야 한다.

테스트는 소프트웨어 공학에 있어서 유지보수 비용과 관련하여 점점 중요성이 높아지고 있는 분야이며, 이는 SOA를 적용한 플랫폼에서도 마찬가지이다. 따라서 SOA를 위한 빠른 적응력과 민첩성을 충족 할 수 있는 테스트 기법에 대한 체계적인 연구가 필요하다. SOA에서의 테스트는 고전적인 V-모델에서의 시스템 테스트, 인수 테스트와 같은 개발 후반부의 테스트보다 단위 테스트와 같은 설계와 병행되는 개발 초반의 테스트에 더욱 비중을 두어야 한다. 이는 애자일(Agile) 방법론에서 제창하고 있는 빠른 적용과 테스트, 그리고 릴리즈 이론과 맥을 같이하며 이를 통해 환경의 변화에 민첩하게 적응할 수 있다. 그러한 이유로 본 논문에서는 SOA에서의 단위 테스트를 위해서 기본 로직 단위인 '서비스'를 테스트의 단위로 하고, 이 단위의 테스트케이스 생성 기법을 제안한다. 이를 이용하여 서비스 단위의 수정이나 신규 구현을 즉각 테스트하여 릴리즈를 앞당기고 이를 통해 민첩성과 빠른 적응력을 높이게 한다.

본 논문에서는 기존의 테스트케이스 생성 기법을 활용하여 적용시켰다. SOA는 기존의 컴포넌트 나 객체지향 기반의 개발과는 관점의 차이가 있기 때문에 절차나 기법의 접근 방법이 다를 수 있다. 따라서 서비스의 하위 계층으로 내려가 기존 연구를 선택하여 활용 및 적용시킨다. 그리고 해당 계층에서 생성된 결과물들을 통합하여 서비스 하나의 테스트 케이스 결과물로 만든다. 사례적용에서는 인터넷 개인뱅킹 시스템을 통해 제안한 기법의 활용과 적용을 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를, 3장에서는 SOA를 위한 테스트케이스 생성 기법을 소개하고, 4장에서는 이를 이용한 사례적용을 보인 후, 5장에서는 결론으로 마무리 한다.

2. 관련연구

2.1 SOA 에서의 테스트

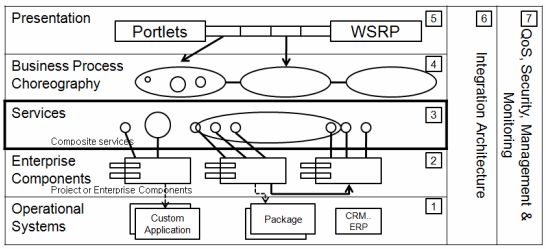
SOA는 로직 단위인 '서비스(Service)' 각각의 사이에 느슨한 결합(loosely coupled)을 한다는 정의를

* 이 연구에 참여한 연구자는 '2 단계 BK21 사업'의 지원을 받았음.

가지고 있다. 로직의 개별 단위들이 상호 간에 서로 고립되지 않고 상호작용하면서도 자율적으로 존재해야 한다. 로직 단위는 여전히 원칙을 준수해야 하지만, 공통성과 표준성을 유지하면서도 독립적으로 진화할 수 있어야 된다. 서비스는 특정한 기준에 따라 로직을 캡슐화한 것이다[2]. 따라서 서비스사이의 테스트는 중요하지 않으며, 서비스 하나의 개발, 수정 그리고 진화에 따른 서비스 단위 테스트가 중요시 된다. 또한 SOA는 매우 비즈니스 친화적(business friendly)이다. 이 말은 기술(technology) 중심이 아니라 SOA가 비즈니스에 의해 운영(business-driven)된다는 것이다. 그렇기에 객체지향 테스트와 컴포넌트 테스트 같은 기존의 테스트 기법을 한 단계 정제하여 하여 테스트를 시행해야 한다.

SOA를 위한 개발 방법론은 관점에 따라서 여러 연구가 이루어 졌다. SODA(Service Oriented Development of Applications)[3], SOUP(Service Oriented Unified Process)[4], SOAD(Service Oriented Analysis and Design)[5], 그리고 SOMA(Service-Oriented Modeling and Architecture)[6]와 같은 방법론이 대표적이다.

이중 SOMA는 서비스 개발 단계를 서비스 식별, 명세, 실현 세 단계로 구성했으며, SOA의 구조를 계층구조로 정의하였다. 그림1은 SOMA의 계층구조를 보여주고 있으며, 상위 계층으로부터 해당되는 하나 또는 여러 개의 서비스조합으로 식별하고, 서비스는 또한 컴포넌트로 구체화된다. 이 컴포넌트는 기존 어플리케이션 리소스(legacy system)로 이루어지는 계층을 가진다.



(그림 1) SOMA 에서의 서비스 계층

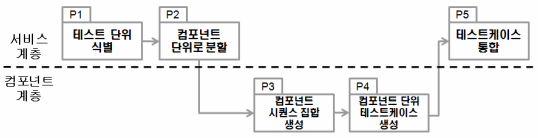
2.2 테스트케이스 생성 기법

기존의 객체지향 개발과 컴포넌트 기반 개발에서의 테스트케이스 생성 기법은 많은 연구가 되어 있는 상태이다. UML을 이용한 연구로는 사용사례(Use Case)를 기반으로 하거나, 상태 다이어그램(State Diagram)이나 협력도(Collaboration Diagram)를 이용한 테스트케이스 생성 기법이 있다[7][8]. Object-Z나 OCL을 이용한 테스트케이스 추출 기법, 메시지 패스를 이용한 확장사례 기반 테스트케이스 추출기법과 같은 블랙박스 테스트 기법 연구도 있다[8]. 상호운용성을 위한 테스트케이스 생성 기법[9]은 특히 테스트케이스를 만들 때의 입력이 되는 EFSM명세를 생성하는 기법을 구체화 시켰다.

테스트케이스 생성에 있어서 가장 빈번히 사용되는 것이 EFSM(Extended Finite State Machines)이다. 어떠한 시스템의 유한개의 상태를 모델링하기 위해서 FSM(Finite State Machines)이 많이 사용되어 왔다. FSM의 부족한 점을 추가해서 만들어진 것이 EFSM으로 행위(action)를 추가하여 상태전이를 나타내어 주며, 입력심볼, 출력심볼, 상태, 변수, 전이의 5개의 튜플로 이루어진다.

3. SOA 를 위한 테스트케이스 생성 기법

3장에서는 SOA를 위한 서비스 단위의 테스트 케이스 생성 기법에 대해서 기술한다. 그림2는 제안 기법에 대한 도식화이다. P1에서 테스트 할 서비스를 선정하고, P2에서 서비스를 컴포넌트 단위로 분할한다. P3에서 하위 계층으로 분해된 컴포넌트들을 정렬하여 컴포넌트 시퀀스 집합을 생성한다. P4에서는 컴포넌트 각각의 테스트케이스를 생성한다. 마지막 P5에서 모든 컴포넌트들의 테스트케이스들을 컴포넌트 시퀀스 집합과 합성하여 하나의 서비스 테스트 케이스를 만들게 된다.



(그림 2) SOA 를 위한 테스트케이스 생성 기법

3.1 테스트 단위 식별(P1)

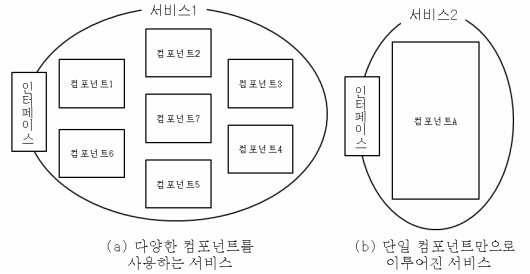
SOA에서의 '서비스'는 비즈니스 친화적 개념이며 주관적인 단위로 인식되고 있다. 때문에 이러한 서비스의 범위를 판단하기 위해 '서비스 식별'이라는 주제의 연구가 활발히 진행되고 있다. 본 연구에서는 개발자의 관점에서 서비스를 바라보며, 따라서 식별의 대상이 아니라 이미 식별되어 있고 설계가 완료된 서비스를 테스트 목표로 삼는다. 또한 SOA의 느슨한 결합이라는 특성에 따라 각 서비스간의 상호운영성은 다루지 않으며, 단일 서비스를 테스트 단위로 정한다. 즉 P1은 설계가 완료되어 하부 계층을 이루는 컴포넌트까지 구현된 하나의 서비스를 테스트 목표로 선정하는 단계이다.

3.2 컴포넌트 단위로 분할(P2)

본 연구에서는 기존의 테스트케이스 생성 기법을 적용하기 위해서, 테스트의 목표로 선정된 서비스를 하위 계층으로 분할한다. 주관에 의해 범위가 정의된 서비스라 할지라도 그 하부 계층은 기존 연구를 적용시킬 수 있는 컴포넌트 혹은 기존 어플리케이션 리소스(legacy system)로 이루어져 있다[6]. P1에서 선택된 서비스를 하위 계층의 단위로 나눈다. 이때 본 연구에서는 서비스의 하위 계층 중 컴포넌트 계층을 사용한다. 서비스 설계로부터 목표 서비스에서 사용된 컴포넌트들을 선정하여 나열한다.

그림3은 서비스와 그를 이루는 컴포넌트들의 예이다. 하나의 서비스는 여러 개의 컴포넌트로

이루어져 있으며, 서비스 식별의 관점에 따라 하나의 컴포넌트만으로 이루어 졌을 수도 있다.



(그림 3) 서비스를 이루는 컴포넌트

3.3 컴포넌트 시퀀스 집합 생성(P3)

서비스의 하위 계층으로 사용되는 컴포넌트들은 서로간에 데이터를 주고 받으며 영향을 미치게 된다. 서비스와 서비스 사이의 개념인 느슨한 결합과 달리 서비스 하부 계층인 컴포넌트와 컴포넌트간의 결합은 강한 결합이 된다. 그렇기에 서비스 하나의 테스트케이스 생성을 위해 그 하부 컴포넌트들을 연결 관계에 따라서 정렬하여야 한다.

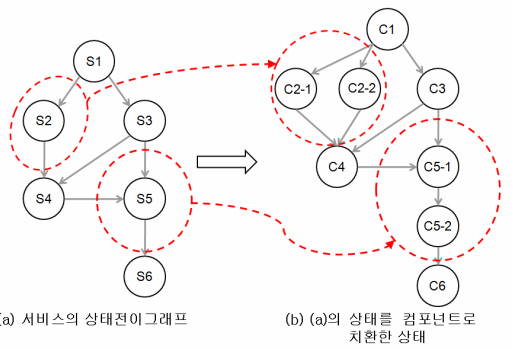
먼저 P1에서 선정된 서비스에 대한 상태전이 그래프를 그린다. 그리고 각 서비스의 상태마다 사용되는 컴포넌트들을 찾아 치환한다. 치환 알고리즘은 다음과 같다.

서비스 상태를 컴포넌트로 치환 알고리즘

```
서비스의 상태전이그래프 G 에서

1 각 서비스 상태에서
2     해당 상태에서 사용되는 컴포넌트 선택
3         컴포넌트 개수 > 1 인 경우
4             모든 컴포넌트에 대하여
5                 동시에 실행되는 컴포넌트는 병렬 정렬
6                 선후 관계 컴포넌트는 직렬 정렬
7     해당 상태를 정렬된 컴포넌트 집합으로 치환
8     추가된 컴포넌트들을 기존의 상태나 컴포넌트와
      연결
```

그림4는 임의의 서비스의 상태전이그래프에서 각 서비스 상태 노드가 컴포넌트로 치환되는 그림이다. (a)의 각 노드는 서비스의 상태를 나타내고, (b)의 각 노드는 컴포넌트를 나타낸다. 위쪽 점선 부분은 S2 상태를 컴포넌트로 치환하는 그림이다. 사용되는 컴포넌트가 C2-1, C2-2 두 개이고, 이 두 개가 동시에 기능하는 컴포넌트들이기 때문에 병렬 형태로 치환한다. 컴포넌트간의 선후관계가 있는 경우에는 아래쪽 점선 부분과 같이 직렬 형태로 치환한다 치환 후에는 기존의 컴포넌트나 상태와 연결한다.



(그림 4) 서비스 상태를 컴포넌트로 치환

치환된 그래프를 다시 <표1>의 컴포넌트 시퀀스 생성 절차를 걸쳐 서비스의 컴포넌트 시퀀스 집합을 만든다. Step3부터 Step5까지는 [9]의 비순환 경로 검색 단계를 참고하였다.

<표 1> 컴포넌트 시퀀스 집합 생성 절차

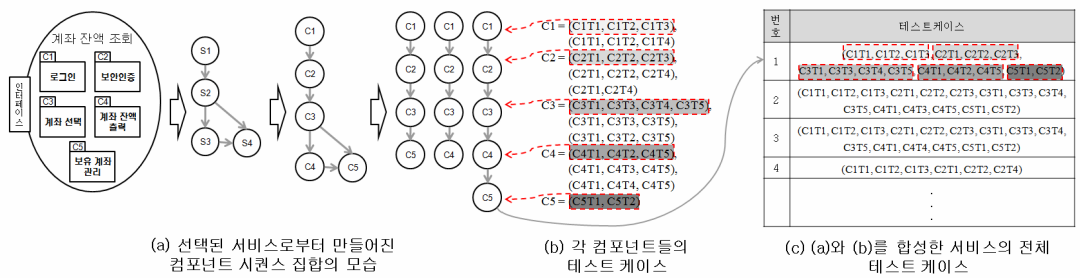
Step	설명
1	서비스의 상태전이그래프 생성
2	Step1의 결과물을 각 상태에서 사용되는 컴포넌트로 치환
3	Step2의 결과물을 반복되는 정점이 없는 모든 비순환 경로로 만듦
4	소규모의 다른 사이클을 포함하지 않는 가능한 모든 단순 사이클 생성
5	Step3의 경로들과 사이클들을 최종 경로 집합과 결합

3.4 컴포넌트 단위 테스트케이스 생성(P4)

P3에서 정렬된 컴포넌트들은 각각 개별적인 컴포넌트 단위로써의 테스트케이스 생성을 거쳐야 한다. 이때의 테스트케이스 생성은 EFSM 명세를 만들고 이를 입력으로 하여 테스트케이스를 생성하는 기존의 기법[9]을 활용한다. 이를 통해 컴포넌트간의 상호운용성이 충족 된다..

3.5 테스트케이스 통합(P5)

마지막 과정인 테스트케이스 통합은 P3에서 생성된 컴포넌트 시퀀스 집합에 P4에서 만들어진 각 컴포넌트의 테스트케이스를 합성하는 단계이다. 컴포넌트마다 여러 테스트케이스를 가지고 있으므로 각 상태에서 사용되는 테스트케이스를 시나리오에 맞게 선정하여 치환한다. 컴포넌트의 테스트케이스가 2개 이상인 경우가 빈번하므로 치환 시에 다양한 조합이 나타나며, 시나리오에 맞는 조합을 모두 만들어 내야 한다. 예를 들어 C1과 C2가 직렬 형태로 이루어진 컴포넌트 시퀀스 집합이 있을 때, 각각 C1이 4개, C2가 3개의 컴포넌트 테스트케이스 집합을 가지고 있다면 최대 12개의 조합이 가능하다.



(a) 선택된 서비스로부터 만들어진 컴포넌트 시퀀스 집합의 모습 (b) 각 컴포넌트들의 테스트 케이스 (c) (a)와 (b)를 합성한 서비스의 전체 테스트 케이스

(그림 5) 계좌잔액조회 서비스 테스트케이스 생성

시나리오에 따라 사용되지 않는 경우를 제외하고는 모두 치환하여 통합한다.

4. 사례적용

이 장에서는 3장의 기법을 인터넷 개인 뱅킹 시스템의 계좌잔액조회 서비스에 적용하여 case study를 한다. 그림5의 (a)는 계좌잔액조회라는 서비스가 5개의 컴포넌트로 이루어져 있으며, 이것이 4가지의 상태로 변화하며 S1, S2, S3, S4의 노드로 표현 되었다. 그리고 각 노드를 해당 상태에서 사용되는 컴포넌트로 치환한다. 특히 S1노드에서는 컴포넌트 C1과 C2가 사용되며 각 컴포넌트의 선후관계를 따져 직렬 형태로 치환하였다. 모든 노드를 컴포넌트로 치환한 후에 컴포넌트 정렬 절차를 걸쳐 3개의 컴포넌트 시퀀스 집합을 얻었다. (b)는 5개의 컴포넌트 각각의 테스트케이스이다. C1, C2, C3, C4, C5는 컴포넌트이며 각 테스트케이스들은 구분을 위하여 'C(컴포넌트번호)T(테스트케이스번호)'로 표기하였다. (a)에서 나온 컴포넌트 시퀀스 집합의 컴포넌트 노드들을 (b)의 테스트케이스들로 치환하여 단일 서비스의 테스트케이스를 얻게 됨을 볼 수 있다.

5. 결론 및 향후 연구

SOA에 대한 관심과 사용이 늘어가면서, 체계적이며 구체적인 기법 연구가 요구되고 있다. 특히 테스트는 유지보수의 문제로 제정적인 문제에 큰 영향을 미치고 있으며, 이는 비즈니스 관점에서 운영되는 SOA에 있어서 더욱 부각된다. 따라서 SOA에 알맞은 테스트 기법에 대한 연구가 요구되며, 본 연구에서는 SOA를 위한 테스트케이스 생성 기법을 제안하였다. 서비스를 컴포넌트 단위로 분할하고 이를 정렬한 후에, 각각의 컴포넌트에 기존의 테스트케이스 생성 기법을 적용하여 컴포넌트간의 상호운용성을 높인다. 그리고 각각의 테스트케이스를 종합하여 하나의 서비스단위 테스트 케이스가 만들어진다. 이러한 절차를 통해서 기존의 테스트케이스 생성 기법을 SOA에 알맞게 변형, 사용하여 SOA에 대한 적합성을 높였다. 본 연구의 제안 방식을 통하여 SOA에서의 서비스 단위 테스트를 체계적으로 수행할 수 있게 된다. 특히 서비스의 구현 혹은 수정 후에 즉각적으로 테스트가 가능하고, 이는 빠른 릴리즈를 실현한다. 이를 통해 실시간 기업을 위한 빠른 적응력과 민첩성 같은 요구사항을 충족 할 수 있게 된다.

향후 연구로는 서비스 계층과 하부 계층 사이의 분할과 조합시의 구체적인 식별 기법에 대한 연구가 요구된다. 또한 본 연구의 자동화 구현과 함께 효율성 증명이 필요하다.

참고문헌

[1] 백종현, 김형석, 김영호, 한상인, "SOA 플랫폼 분석과 시장전망", 한국정보과학회, 정보과학회지 제25권 제1호, 2007. 2

[2] Erl, Thomas, Service-oriented Architecture: Concepts, Technology, and Design, Prentice Hall PTR, 2005.

[3] Gregg Kreizman, "How to Build a Business Case for Service-Oritented Development of Applications in Government", Gartner Industry Research, Sep. 2005

[4] Kunal Mittal, "Service Oriented Unified Process(SOUP)", IBM Journal, Jun. 2005

[5] Ash Parikh, Rajesh Pradhan, Nirav Shah, "Modeling of Web Services : A Standards-Based Approach", Software Magazine, May. 2004

[6] Ali Arsanjani, "Service-oriented modeling and architecture : How to identify, specify, and realize services for your SOA", IBM developerWorks, Nov. 2004

[7] Kim, Y.G. Hong, H.S. Bae, D.H. Cha, S.D., "Test cases generation from UML state diagrams", Software, IEEE Proceedings, Vol.146, Issue.4, Oct. 1999

[8] 서광익, 최은만, "객체지향 소프트웨어를 위한 주요 블랙박스 테스트 기법들의 비교", 한국정보과학회, 정보과학회논문지 : 소프트웨어 및 응용 제33권 제1호, pp. 1-17, 2006. 1

[9] 이지현, 노혜민, 유철중, 장옥배, 이준욱, "상호운용성 테스트를 위한 테스트케이스 생성 기법", 정보과학회논문지 : 소프트웨어 및 응용 제33권 제1호, pp. 44-58, 2006.1