DOI: 10.3745/KIPSTD.2009.16-D.4.595

BPMN기반의 모델 단축을 이용한 서비스 시스템의 테스트 케이스 생성 기법

이 승 훈[†] · 강 동 수[††] · 송 치 양[†††] · 백 두 권[††††]

요 약

시스템 개발에서 초기 테스트는 오류수정 비용을 현저하게 낮출 수 있으며 이는 SOA기반 서비스 시스템에서도 여전히 중요한 요소이다. 그러나 서비스 시스템을 위한 기존 테스트 케이스 생성 기법들은 XML 기반 언어를 사용하여 웹서비스에 한정적이라는 한계점을 가진다. 이를 극복하기 위해서 본 논문에서는 BPMN에 따라 작성된 비즈니스 프로세스 기반으로 서비스 시스템의 테스트 케이스 생성 기법을 제시한다. 테스트 노력을 최소화 하기 위해 기존 BPM을 워크플로우의 기본 요소로만 단순화 시킨 S-BPM을 사용한다. 테스트 케이스 생성 과정은 목표 서비스 시스템에 대해 S-BPM을 생성하고, 이를 방향성 그래프로 변환 후, 시나리오 검색 알고리즘을 적용하여 서비스 시나리오를 생성하고, 메시지의 이동 정보를 추출함으로써 이루어진다. 본 기법을 적용하면 웹서비스에 한정적이지 않으면서도 범용 서비스에 적용이 가능한 효율성 높은 테스트 케이스를 얻을 수 있으며, 이 결과물은 SOA의 비즈니스 관점 지향 특징을 반영한 테스트 케이스로 볼 수 있다.

키워드 : 서비스 시스템, 테스트 케이스, 비즈니스 프로세스, BPMN, SOA

A Method of Test Case Generation using BPMN-based Model Reduction for Service System

Seung-Hoon Lee[†] · Dongsu Kang[††] · Cheeyang Song[†††] · Dookwon Baik[††††]

ABSTRACT

The early test can greatly reduce the cost of error correction for system development. It is still important in SOA based service system. However, the existing methods of test case generation for SOA have limitations which are restricted to only web service using XML. Therefore, this paper proposes a method of test case generation using BPMN-based model reduction for service system. For minimizing test effort, an existing BPM is transformed into S-BPM which is composed of basic elements of workflow. The process of test case generation starts with making S-BPM concerning the target service system, and transforms the target service system into directed graph. And then, we generate several service scenarios applying scenario searching algorithm and extract message moving information. Applying this method, we can obtain effective test cases which are even unlimited to web service. This result is the generation of test case which is reflected in the business-driven property of SOA.

Keywords : Service System, Test Case, Business Process, BPMN, SOA

1. 서 론

SOA(Service-Oriented Architecture)는 최근 급부상한 패러다임으로 많은 주목을 받고 있다[1]. 특히 발전된 웹서비스 기술과 접목되면서 경쟁력을 극대화하기 위해 빠른 적응력과 민첩성을 목표로 하는 실시간 기업(Real Time Enterprise)에게 가장 적합한 정보기술 아키텍처로서 각광받고 있다[2]. SOA는 전통적인 개발방법과 여러 차이점을 보인다. CBD(Component-Based Development) 방법론이나 객체지향 방법론에서는 데이터 흐름이 중심이며, 각 방법론의 기초 단위인 컴포넌트나 객체는 상태(State)를 가지고 있어 입력에 의해 변화된다. 따라서 테스트 케이스 생성을 위해서 FSM(Finite State Machine)을 입력으로 사용한다. 반면 SOA는 느슨한 결합의 특징을 갖고 있어서 데이터가 아닌 메시지 교환중심의 아키텍처를 가진다. 또한 SOA의 기본 단위가 되는 서비스(Service)는 상태가 없으므로[3] 기존에 사용하던 FSM의 적용이 어려운바, 이를 해결하기 위한

※ 이 연구에 참여한 연구자는 '2단계 BK21 사업'의 지원을 받았음.
본 과제는 한국소프트웨어진흥원의 SW공학 요소기술 개발과 전문인력 양성사업의 결과물임을 밝힙니다.
† 준 회 원 : 고려대학교 컴퓨터·전파통신공학과 석사과정
†† 준 회 원 : 고려대학교 컴퓨터·전파통신공학과 박사과정
††† 정 회 원 : 경북대학교 소프트웨어공학과 조교수
†††† 종신회원 : 고려대학교 컴퓨터학과 교수
논문접수 : 2009년 6월 24일
심사완료 : 2009년 7월 20일

SOA에 적합한 새로운 기법들이 연구되고 있다[4]

테스트는 소프트웨어 공학에 있어 비용과 관련하여 중요하게 여겨지는 분야로 이러한 점은 SOA에서도 유지되고 있다[5]. 그러나 SOA의 특징으로 인하여 고전적 방법론에서의 테스트와 차이점이 발생한다. 특히 기존 방법론은 코드 중심의 개발자 관점 테스트였다면 SOA에서의 테스트는 비즈니스 관점으로 진행되어야 한다. 또한 SOA의 논리 단위인 서비스는 사용자의 입력과 컨텍스트에 따라서 동적으로 재조합되며, 복합 서비스로써 사용자의 이익을 창출한다. 이러한 서비스들이 모여 하나의 서비스 시스템이 되고, 시스템 단위에서의 테스트가 필요해 진다. 시스템 테스트는 Use-Case나 다른 시스템 요구 분석의 결과로부터 생성된 테스트 케이스를 사용하여 사용자의 모든 요구를 하나의 시스템으로서 완벽하게 수행하는지를 테스트하는 것을 의미[6]한다. 이 때 "얼마나 많은 테스트 케이스가 필요한가" 라는 물음은 테스트에 있어서 가장 중요한 문제이다. 너무 적은 테스트 케이스는 버그를 놓치는 경우가 생길 수 있고, 너무 많은 테스트 케이스는 비용소모가 심해진다[7].

이와 같은 문제들을 해결하기 위하여 본 논문은 비즈니스 프로세스를 사용하여 SOA의 특징에 부합하는 테스트 케이스 생성 기법을 제안한다. 이 기법은 SOA의 단위인 서비스들이 모여 이루어지는 서비스 시스템과 그 시스템 내부에서 이루어지는 서비스나 그 하위 단위들 간의 메시지 전송을 테스트 하여 서비스 시스템 내의 재조합이 올바르게 수행되는 것을 확인하는 것을 목표로 한다. 메시지 이동 정보를 얻기 위해 BPMN(Business Process Modeling Notation)의 규칙에 맞추어 생성된 워크플로우 모델인 BPM(Business Process Model)을 이용한다. 모든 서비스는 워크플로우로 나타낼 수 있고, 워크플로우를 통해서 서비스 시스템 내에서 이동하는 메시지들을 파악할 수 있다. 따라서 BPM을 통해 상태가 없는 서비스 시스템 내부의 동작을 인식할 수 있다. BPM은 비즈니스 관점의 요구사항이 표현된 것이므로 테스트 케이스 생성의 입력으로 사용됨으로써 테스트에 비즈니스 관점이 직접적으로 투영되는 효과를 가질 수 있다. 또한 테스트 케이스 생성의 비용 소모를 최소화 하기 위해 기존 BPM 모델을 단순화한다. 이것을 위해 워크플로우의 기본 요소로 이루어진 S-BPM을 제시하고 이를 방향성 그래프로 변경하여 서비스 단순 경로 그래프를 생성한다. 단순 경로 그래프를 대상으로 서비스의 전체 시나리오 검색 알고리즘을 적용하여 효율성이 높은 서비스 시나리오를 생성하고 메시지 이동을 추출하여 서비스 테스트 케이스를 도출한다.

본 논문의 구성은 다음과 같다. 2장에서는 기존 기법들의 문제점을 살펴보고, 본 논문에서 사용된 기법들을 이해한다. 3장에서는 BPMN을 이용한 서비스 시스템의 테스트 케이스 생성기법을 소개한다. 4장에서는 적용 사례를 보이고 5장에서 제안 기법의 효율성을 평가하고, 기존 연구와의 비교를 통해 제안 기법의 신뢰성을 보인다. 마지막 6장에서 결론과 향후 연구 계획을 통해 마무리 한다.

2. 관련 연구

먼저 SOA에서의 테스트 특징을 소개하고 기존 연구의 문제점을 살펴본다. 이후 본 논문의 핵심이 되는 표기법인 BPMN을 소개한다. 또한, 제안 기법의 효율성을 높이기 위하여 사용된 단순 경로 알고리즘을 기술한다.

2.1 SOA에서의 테스트 기법

서비스는 관점에 따라서 다양한 정의를 가지고 있으며 본 논문은 "서비스는 인간의 삶을 보조함으로써 제공자와 수혜자를 위하여 가치를 창조하는 사용자 수준의 워크플로우이다."[8]라는 정의를 사용하고 이 정의에 따르면 서비스는 워크플로우로 표현할 수 있다. 또한 비즈니스 프로세스는 하나 이상의 비즈니스 서비스로 구성되며 BPM이나 ESB (Enterprise Service Bus)에 의해 관리 가능하다. 여기서 비즈니스 서비스는 독립적으로 단위 업무 수행, 제공하는 서비스를 말한다[9]. 위의 정의들을 통해 서비스를 BPM으로 표현하고 관리할 수 있음을 알 수 있다.

서비스 시스템은 서비스를 기술적으로 구현한 단일 서비스 혹은 복합 서비스까지 포함하는 사용자를 위한 하나의 완전할 활동을 의미한다. 완전한 활동이란 사용자의 요청을 받아 해당하는 처리를 거쳐 다시 사용자에게 결과를 알려주는 시작과 끝이 존재하는 일련의 과정을 말한다. 서비스를 제공하는 개체로서 소프트웨어와 하드웨어와 같은 이종 컴포넌트들의 통합으로 시스템이 구성된다. 사용자의 요구에 따라서 원자(Atomic) 서비스나 컴포넌트간에 상호작용을 통해 실행 시간에 동적으로 재구성되어 서비스를 제공하는 특징을 가지고 있다. 본 논문의 목표 테스트 단위는 서비스 시스템이다.

전통적인 개발 방법론들은 분석, 디자인, 구현 그리고 테스트로 이어지는 전형적인 개발 방식을 많이 사용해 왔다. 반면 SOA는 모델링을 거쳐, 컴포넌트나 서비스를 합치는 조립단계를 통해 새로운 서비스를 만들어내고 이를 테스트하고, 동작하는 것을 모니터링하여 다시 모델링에 반영하는 개발 방식을 사용한다[10]. 이때 모델링 단계에서 산출물로 생성되는 것이 비즈니스 프로세스를 나타내는 BPM과 같은 모델들이다. 또한 SOA는 서비스간에 느슨한 결합을 한다는 중요한 특징을 가지고 있다. 이 때문에 서비스 간에 주고받는 메시지가 중요시 되며 메시지가 올바르게 이동하는지 확인하는 것이 SOA에서의 테스트 목적 중 하나가 된다. 반면 서비스 시스템에서 사용되는 원자 서비스나 컴포넌트는 블랙박스로 간주하여 하나의 부품처럼 사용하여 내부의 동작은 테스트 대상에서 제외된다.

기존의 SOA를 위한 테스트 기법들은 Data Perturbation을 이용한 서비스의 종단간 메시지 교환에 집중한 연구[11], WSDL명세서 기반의 테스트 케이스 생성[12], CV&V Framework를 통한 복합 서비스 테스트[13], Decision Table을 사용하여 자동으로 테스트 케이스를 생성하는 연구[14]와 같은 기법들이 존재한다. 기법을 적용할 수 있는 서비스의

범위나 테스트 방법에서 차이를 보이고 있지만 공통적으로 XML기반으로 작성된 언어를 통해서 테스트 케이스를 생성한다. 이것은 FSM을 사용할 수 없는 한계는 극복하였지만, 적용할 수 있는 환경이 웹서비스로 한정된다는 문제점과 전통적인 개발기법의 소스 코드와 비견되는 XML언어를 입력으로 사용하여 여전히 개발자 관점 위주의 테스트가 되어 SOA의 특성을 반영하고 있지 못하는 문제점을 동시에 안고 있다.

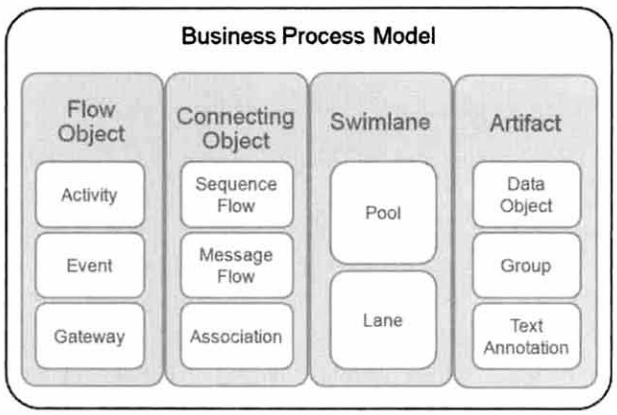
2.2 BPMN

BPMI(Business Process Management Initiative)에서 비즈니스 프로세스 모델에 관한 업계표준을 위해 만든 것이 BPMN이다. 현재는 OMG(Object Management Group)에 의해서 관리되고 있으며 2009년 1월, 현재 버전인 BPMN 1.2까지 나온 상태이며 본 논문에서는 BPMN 1.1버전을 기준으로 한다. BPMN은 비즈니스 프로세스를 올바르게 디자인하고 엔지니어와 실무자들의 원활한 의사소통을 위하여 표준화된 그래픽 표기법을 사용한다. 또한, 이 모델 표기법은 UML의 활동 다이어그램(Activity Diagram)과도 유사한 모습을 보여 많은 비교가 되지만, 다른 어떠한 비즈니스 프로세스 모델링 표기법보다 강력한 표현력과 확장성을 가지고 있다. 특히 서비스를 구현하는 언어인 BPEL(Business Process Execution Language)로 직접적인 변환이 가능하여 SOA환경에서는 최적의 워크플로우 표현기법으로 평가 받고 있다[10][15][22]. BPMN을 사용하여 표현된 워크플로우 모델을 BPM(Business Process Model)이라 한다. BPM은 목표 서비스 시스템이 제공할 수 있는 동작에 대한 비즈니스 관점의 요구사항을 표현한다. 작성자의 표현 관점에 따라서 변경될 수 있으나 서비스 시스템의 자세한 활동부터 전체적인 활동까지 모두 나타낼 수 있으므로 실제 시스템이 BPM의 표현을 만족한다면 성공적인 시스템으로 볼 수 있다. 따라서 BPM을 테스트 케이스 생성의 기준으로 함으로써 비즈니스 관점의 요구사항을 테스트에 투영시킬 수 있으며 시스템의 목표를 확인하기에 용이해진다.

(그림 1)은 BPMN의 기본 요소 및 구성을 나타내는 것[16]으로 흐름객체(Flow Object), 연결객체(Connecting Object), 스윔레인(Swimlane), 부가객체(Artifact)의 네 가지 구성 요소를 가지고 표현한다.

각 요소는 다음을 통해 간략히 소개한다. 흐름객체(Flow Object)는 프로세스에서 특정 행위를 하는 객체이다. 프로세스의 시작과 끝을 나타내거나 프로세스의 진행을 조절하는 Event와 실제 작업을 수행하는Activity, 그리고 프로세스의 흐름을 판단하고 결정하는 Gateway로 구성되어 있다. (그림 2)의 (a)는Activity의 표기법 기호이다. (b)는 Activity의 특수한 경우로 내부에 서브 프로세스가 있음을 표현한다. (c)와 (d)는 각각 AND와 XOR Gateway의 표기법 기호이다.

연결객체(Connecting Object)는 프로세스의 흐름을 표현하는 객체이다. 흐름객체들을 연결하여 프로세스의 진행방향과 순서를 표현하는 Sequence Flow, 다른 Pool과의 통신



(그림 1) BPMN 구성

을 표현하는 Message Flow, 프로세스의 객체들 간의 연관상태를 표현하는 Association으로 이루어져 있다. 그림2의 (e)는 Sequence Flow의 표기법 기호이다.

스윔레인(Swimlane)은 Pool과 Pool안에서 표현되는 Lane으로 구성되어 있다. 이러한 표기를 이용하여 객체들의 역할이나 조직의 구성단위를 표현한다.

부가객체(Artifact)는 프로세스의 흐름정보와는 직접적인 관련이 없으나, 프로세스의 부가적인 정보를 표기하기 위해 사용한다. 프로세스의 데이터들이 어떻게 사용되는지를 표현하는 Data Object, 요소들을 묶어주는 Group, BPM에 추가적인 정보를 제공하는 Text Annotation으로 구성되어 있다.

한편 WfMC(Workflow Management Coalition)에서 판단요소인 AND-Split, AND-Join, XOR-Split, XOR-Join과 연결요소인 Sequence와 특정 조건을 만족할 때까지 특정 범위를 반복 수행하는 Cycle 만을 워크플로우의 기본 구성 요소로 정의하고 있다[17]. BPMN에서는 흐름객체(Flow object)의 Gateway 중에서 AND-Split, AND-Join, XOR-Split, XOR-Join의 4 가지가 기본 구성요소에 해당된다. 연결객체(Connecting Object)중에서 Sequence Flow가 기본 구성 요소 Sequence에 해당되며, Cycle은 Sequence Flow만으로 표현이 가능하다. (그림 2)에서 보이는 기호들은 모두 워크플로우 기본 구성 요소의 BPMN 표기법이다.

(그림 2) 워크플로우의 기본요소에 해당하는 BPMN 요소

2.3 단순 경로 알고리즘

본 논문에서 S-BPM에 대해 서비스 시나리오를 도출하기 위해서 단순 경로 알고리즘이 필요하다. 기존 단순 경로 자료 구조 중 본 논문에서 사용된 DAG, SCC 및 Next-Transition-Tree와 관련 알고리즘을 살펴본다. 본 기법은 목표 서비스 시스템의 S-BPM에서 SCC를 검색하고 DAG를 표현하여 Next-Transition-Tree를 통해 순환경로를 단축하

는 과정을 통해 테스트를 위한 효율적인 서비스 시나리오를 얻는다.

2.3.1 DAG

DAG(Directed Acyclic Graph)란 간선이 방향성을 가지고 있고, 시작과 끝 정점이 같은 순환 경로를 포함하고 있지 않은 그래프를 뜻한다[18]. 본 논문에서는 방향성 그래프상에서 순환 경로를 단순화 시키기 위하여 사용되었다. 이는 순환 경로를 가지고 있는 서비스 프로세스에 대한 테스트 케이스를 최적화하여 생성하기 위함이다. 본 논문에서 DAG는 BPM을 기반으로 만들어진 방향성 그래프에 사용된다. 따라서 DAG의 각 노드는 서비스에서 사용된 여러 개의 컴포넌트들은 나타낸다. 다음은 DAG로부터 전체 비순환 경로를 검색하는 알고리즘이다. (그림 3)의 (b)에 이 알고리즘을 적용하면 비순환 경로 집합 P ={(A1, A2, SCC1, A6), (A1, SCC1, A6))를 얻는다.

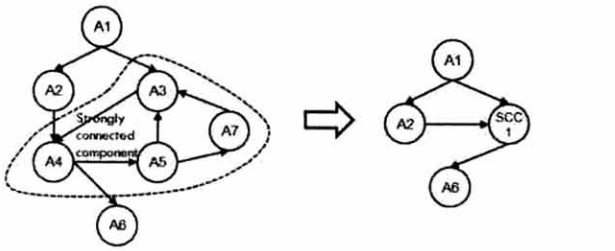
PATHS-IN-DAG 알고리즘

입력. 하나의 소스 s와 여러 개의 말단 간의 DAG인 G'
출력. G'에서 모든 소스-말단 경로
참고. $p(v_i)$ 는 G'에서 모든 v_i-말단 경로를 표현

1. G'의 노드들을 위상 정렬 : $s = v_0, v_1, \ldots, v_{n-1}$;
2. for i=n-1, ..., 0
3. 만약 v_i가 말단 노드이면
4. $p(v_i) = \{\}$;/* 경로가 없는 단일 집합*/
5. 아니면
6. v_i 부터 시작되는 간선을 $w_1, \ldots, w_r$라 하면;
7. $p(v_i) = \bigcup_{j=1}^{r} (v_i, w_j)\, p(w_j)$;
8. return $p(v_0)$

2.3.2 SCC

SCC(Strongly Connected Component)는 순환 경로를 가지고 있는 방향성 그래프에서 DAG를 얻기 위해서 탐색해야 하는 자료구조이다. SCC에서 강연결이란 임의의 방향성 그래프 V(G)에서의 정점의 쌍 v_i, v_j에 대하여 v_i에서 v_j로의 경로, 그리고 v_j에서 v_i 로의 경로가 모두 존재 할 때를 말하고, 방향성 그래프에서 강한 연결 요소는 강연결된 최대 부분 그래프를 말한다. 방향성 그래프에서 SCC를 삭제하면 DAG가 되는데, (그림 3)의 예는 임의의 방향성 그래프에서 SCC를 찾아낸 것을 (a)에서 보여주고, 그 SCC를 축약하여 DAG로 만든 것을 (b)에서 보여준다.



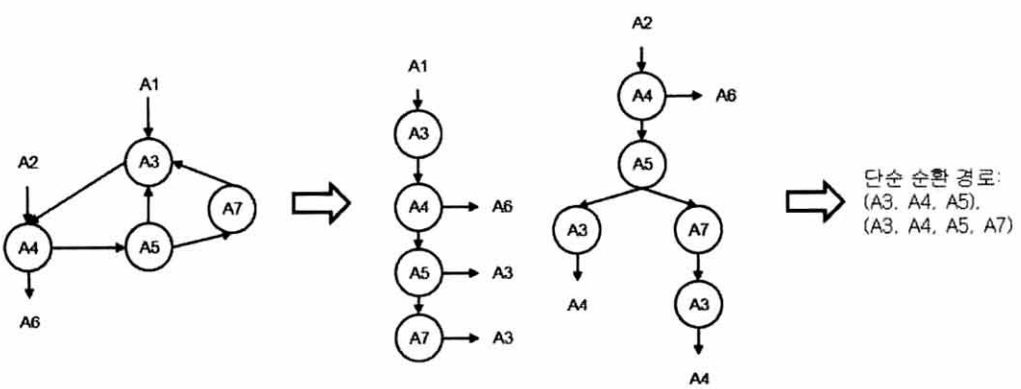
(a) 임의의 방향성 그래프 (b) (a)에서 SCC가 축약되어 만들어진 DAG

(그림 3) 방향성 그래프와 DAG

2.3.3 Next-Transition-Tree

Next-Transition-Tree[19]는 단순 순환 경로를 찾기 위한 간단한 자료구조이다. 임의의 방향성 그래프에서 노드 v가 존재할 때, Next-Transition-Tree T(v)는 노드 v로부터 다른 노드까지의 모든 비순환 경로를 나타낸다. 이 자료구조는 그래프에서 어떠한 노드에도 적용될 수 있다. 트리구조에서 v는 루트노드가 되며, v의 자식노드는 v로 부터 시작되는 방향성 간선을 가지는 모든 노드이다. 일반적으로 어떠한 노드 u의 자식 노드는 u로부터 시작되는 방향성 간선을 가지는 모든 노드이다. 하지만 자식 노드가 루트 노드 v로부터 u까지의 경로에 나타난다면 해당 노드는 트리에 포함되지 않는다. Next-Transition-Tree 에서 노드는 여러 번 나타날 수 있지만, 트리 경로는 그렇지 않다.

본 논문에서는 그 특성상 SCC 내에서만 Next-Transition-Tree가 나타나며 DAG와 연결된 시작 노드들만을 기준으로 Next-Transition-Tree를 생성하면 된다. 이 자료구조를 통해서 단순 순환 경로 집합을 찾을 수 있다. 단순 순환 경로란 더 이상 작은 순환 경로를 가지고 있지 않은 최소 부분 경로를 말한다. (그림 4)는 (그림 3) 에서 찾아낸 SCC에서 시작노드인 A3와 A4에 해당하는 Next-Transition-Tree를 생성한 모습으로 해당 트리를 통해 단순 순환 경로 2개를 찾아 낼 수 있다.



(그림 4) SCC에서 만들어진 Next-Transition-Tree와 이를 통해 생성된 단순 순환 경로 집합

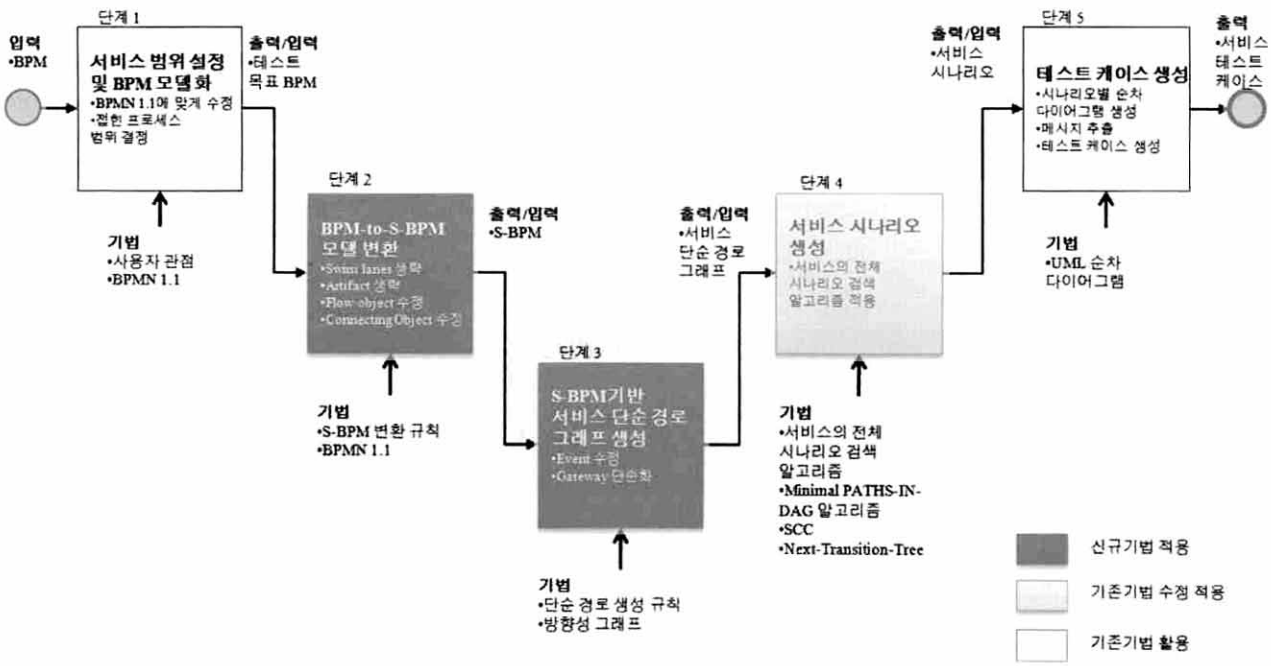
3. 서비스 시스템의 테스트 케이스 생성 기법

본 장에서는 S-BPM(Simple-BPM)을 이용한 서비스 시스템의 테스트 케이스 생성 기법을 수행 절차에 따라서 기술한다. S-BPM이란 일반적인 BPM을 워크플로우의 기본 요소로만 단축한 것이다. (그림 5)는 본 기법의 전체 프로세스를 IDEF0(Integration Definition for Function Modeling 0) 표기법에 의해 표현한 것으로 BPMN기반의 모델 단축을 이용한 서비스 시스템의 테스트 케이스 생성 기법의 프로세스이다. 각 활동의 왼쪽 화살표는 입력을 뜻하며, 오른쪽 화살표는 출력을 뜻한다. 하부 화살표는 해당 프로세스에 영향을 미치는 기법을 나타내며 알고리즘, 자료구조, 기존연구 등이 포함된다.

본 논문은 BPMN 규칙을 준수하지만, 메시지 경로 정보 표현 위주로 간략하게 작성된 S-BPM을 서비스의 비즈니스 모델로 사용한다. 먼저, S-BPM을 만들기 위해 테스트 목표 서비스 시스템에 대한 BPM을 생성한다. 서비스는 사용자와 컨텍스트에 따라 다양한 워크플로우를 가질 수 있다. 본 기법은 이러한 워크플로우를 표현한 BPM을 통해 테스트 케이스를 생성하여 비즈니스 관점이 테스트 케이스 생성에 반영되도록 하고, 메시지 흐름을 중심으로 서비스 테스트를 하는 것이다. 초기 BPM이 생성되면 제시 기법에 따라서 워크플로우의 기본 요소들만 유지하고 기타 요소들은 단축하여 S-BPM으로 변환한다. S-BPM은 일부 정보가 단축되었을 뿐 여전히 BPMN을 따르고 있기 때문에 효율성을 높이기 위해서는 BPM에 적용 가능한 알고리즘을 새로이 연구해야 한다. 하지만 기존의 풍부한 알고리즘들을 활용하는 것이 다양한 대안을 제공하고 비용적으로도 유익하므로 S-BPM을 기반으로 방향성 그래프를 생성한다. 방향성 그래프에 기존 자료구조와 알고리즘들을 수정 적용하여 서비스 시나리오를 생성한다. 이 서비스 시나리오들은 목표 서비스 시스템에서 나타날 수 있는 모든 경로를 최소한 1 회씩 방문하면서도 최단거리로 이루어진 효율성 높은 시나리오들이다. 서비스 시나리오마다 다시 순차 다이어그램을 작성하고 메시지 이동을 추출하여 최종적으로 서비스 시스템의 완성된 테스트 케이스 집합을 얻을 수 있다. 이러한 과정에 대한 세부 절차는 다음과 같다. 첫째, 테스트 목표 서비스의 범위를 설정하고 BPM 모델화한다. 둘째, 보통의 BPM을 워크플로우의 기본 요소로만 이루어진 S-BPM으로 간략화한다. 셋째, 여전히 BPMN의 규칙을 준수하는 S-BPM을 기반으로 방향성 그래프를 생성한다. 넷째, 만들어진 방향성 그래프에 기존의 자료구조 알고리즘들을 수정 적용하여 효율성 높은 서비스 시나리오를 생성한다. 마지막으로 서비스 시나리오에서 테스트 케이스를 추출한다. (그림 5)에서 단계 2의 BPM-to-S-BPM 모델 변환과 단계 3인 S-BPM기반 서비스 단순 경로 그래프 생성이 본 기법의 핵심 부분이다.

3.1 서비스 범위 설정 및 BPM 모델화(단계 1)

본 단계에서는 테스트 목표 서비스 시스템의 테스트 범위를 결정하고 이에 대한 초기 BPM 을 작성한다. 테스트의 목표가 되는 것은 조립 단계 후의 테스트 단계에 위치되는 서비스 시스템이다. 목표 서비스 시스템의 워크플로우는 검증이 완료되어 오류가 없는 BPM을 사용함을 전제로 한다. 이러한 BPM을 작성하는 방법[10]과 오류를 판단하고 수정하는 기법에 관한 연구[16] 역시 많이 진행되고 있으므로 본 논문에서는 다루지 않는다. SOA 개발방식 초기의 모델링 단계에서 서비스를 식별하고 식별된 시스템의 워크플로우를



(그림 5) BPMN기반의 모델 단축을 이용한 서비스 시스템의 테스트 케이스 생성 기법

표기한 BPM을 본 기법에서 사용함을 기본으로 한다. 서비스를 식별하는 것 역시 기존 연구[1]가 진행되고 있는 상태이다. 만약 시스템의 일부분만을 테스트하기 원할 경우에는 서브 프로세스를 단위로 하여 단위별 BPM을 대상으로 선택하면 된다. 앞선 과정을 통해, 요구사항이 테스트의 직접적인 판단 기준이 되어 비즈니스 관점이 테스트에 반영되는 효과를 기대할 수 있다. 만약 요구사항의 수정이나 컴포넌트 조립시의 제한상황으로 인해 초기의 BPM이 변경되었을 경우, 본 단계에서 그에 맞도록 수정하여 사용하도록 한다. 동시에 본 기법에서 사용하는 BPMN 1.1의 표준 표기법을 준수하는 것에 유의한다.

워크플로우의 활동(Activity)은 서비스 내에서 수행되는 단일 작업을 뜻하며 각 활동들은 최소 Component 수준의 표현범위를 가지고 있어야 한다. BPMN은 활동 내부에 서브 프로세스를 포함하고 있는 접힌 프로세스(Collapsed Process)를 요소로 가지고 있으며 (그림 1)의 (b)의 기호로 표현된다. 서브 프로세스는 동기적, 비동기적으로 실행될 수 있으며, 같은 패키지의 범위 내에서 선언되어 다른 프로세스로 상속받을 수도 있다. 접힌 프로세스를 하나의 원자 서비스로 볼 수 있으며 본 단계에서 서브 프로세스를 테스트의 범위에 포함시킬 것인지 여부를 결정한다. 서브 프로세스를 포함시킬 경우에는 해당 활동을 서브 프로세스로 치환하고, 반대의 경우에는 접힌 프로세스를 하나의 활동으로 취급한다.

3.2 BPM-to-S-BPM 모델 변환(단계2)

기존의 BPMN은 풍부한 워크플로우 표현이 가능한 장점을 가지고 있다 그러나 초기 BPM을 통한 서비스 시스템의 테스트 케이스 생성은 복잡하고 많은 테스트 케이스를 생성하여 효율적이지 않다. 본 논문의 목적인 효율성 높은 테스트 케이스 생성을 위해서는 BPMN의 다양한 정보 표현은 오히려 메시지 흐름 정보 추출을 위한 알고리즘의 복잡화를 가져오는 방해요소로 작용한다. 그래서 본 기법에서는 초기 BPM을 경로 정보 손실 없이 최소 표현한 S-BPM(Simple-Business Process Model)으로 변환을 한다. 다시 S-BPM을 기반으로 방향성 그래프를 생성하고 테스트의 시나리오 수를 줄이는 알고리즘을 적용함으로써 테스트 비용을 줄일 수 있다. 이번 절을 통해서 S-BPM을 정의를 기술하고, 초기 BPM을 S-BPM으로 변환하는 규칙을 상세하게 설명한다.

먼저 다음은 S-BPM에 대한 정의이다.

[정의 1 S-BPM] S-BPM은 서비스 시스템 테스트를 위해 BPMN표기법을 간략화한 비즈니스 프로세스 모델이다. 그 구성은 다음과 같다.

가) Activity: 실제로 동작해야 할 일의 단위, Activity는 최소한 하나의 Component로 이루어짐

나) Gateway: 워크플로우에서 분기점에서 다음 활동을 결정하는 지점, 조건에 따라서 다음 활동을 결정하는 판단, AND-Join, AND-Split, XOR-Join, XOR-Split의 종류가 있음

A. AND-Join: 둘 이상의 병렬 프로세스가 공통 제어 쓰레드에 모이는 워크플로우 지점

B. AND-Split: 하나의 공통 제어 쓰레드가 동시에 실행되는 둘 이상의 병렬 프로세스로 나누어지는 워크플로우 지점

C. XOR-Join: 둘 이상의 프로세스가 공통 제어 쓰레드에 모여 하나의 프로세스 결과만 선택하여 다음 프로세스를 진행하는 워크플로우 지점

D. XOR-Split: 하나의 제어 쓰레드가 다양한 프로세스 분기중 다음 활동 선택이 필요할 때 하나를 결정하는 워크플로우 지점

가) Sequence Flow: 워크플로우 상에서 하나의 활동에서 다음 활동으로 진행되는 순서

나) Event: 워크플로우 내에서 특정 조건이 발생, Start, Intermediate, End의 세가지로 구성

본 단계의 변환 규칙은 이전 단계를 통해 정제된 목표 BPM을 S-BPM으로 변환시키게 되며, S-BPM은 BPMN에서 워크플로우를 이루는 기본 요소인 AND, XOR, Sequence에 해당되는 요소만으로 구성된 단축된 BPM이다. 다음 <표 1> 은 그 변환 규칙으로, BPM의 Gateway 요소에 해당 규칙을 적용하여 변환작업을 수행한다.

〈표 1〉 BPMN 요소의 S-BPM 변환 규칙

BPMN 요소		변환규칙	참고
Flow Object	Gateway	AND Gateway 와 XOR Gateway 의 조합으로 표현되도록 변환	• 일반 Gateway는 XOR Gateway로 변경 • AND Gateway 와 XOR Gateway는 유지 OR Gateway나 Complex Gateway는 동일 결과를 나타내는 AND Gateway 와 XOR Gateway 의 조합으로 변환
	Event	None Event로 변환	-
	Activity	유지	
Connecting Object	Sequence Flow	유지	Gateway 변환으로 신규 Sequence Flow 생성 가능
	Message Flow	삭제	-
	Association	삭제	-
Swimlane		삭제	-
Artifact		삭제	-

Flow Object는 3가지 요소마다 다른 규칙이 적용된다. Gateway의 경우 모두 AND와 XOR로만 표현되도록 변환한다. 아무런 표시가 없는 일반 Gateway는 XOR Gateway로 변환한다. OR Gateway나 Complex Gateway는 AND와 XOR간의 조합으로 동일한 정보를 나타내도록 변환한다. 이것은 차후 방향성 그래프 생성시에 규칙을 단순화하기 위한 선행 작업이다. Event들은 동작 조건이 표현되지 않는 None Event로 변환한다. 왜냐하면 Event는 워크플로우의 경로의 조건에 따라서 진행 유무를 표현하는 것이지만 조건 정보는 테스트 케이스 생성에 영향을 미치지 않으므로 조건 정보만 생략하는 것이기 때문이다. Activity는 실제로 테스트 케이스의 경로를 나타내는 정보가 되므로 그대로 유지한다. Connecting Object는 Sequence Flow를 제외하고는 모두 생략하지만 Sequence Flow는 경로 정보를 표현하는 가장 중요한 정보이므로 모든 정보를 유지한다. 단, Gateway들이 변환되면서 Sequence Flow가 추가될 수 있다. Message Flow는 다른 워크플로우와의 메시지 교환 정보이므로 본 논문의 범위에서 벗어나 생략한다. Association은 관계 정보이며 메시지 경로 정보와는 관계가 없으므로 역시 생략한다. Swimlane은 사용자의 이해를 돕기 위한 정보를 표현하지만 메시지 경로에 해당하는 정보는 가지지 않으므로 생략하고, Artifact 역시 같은 이유로 생략한다. 이러한 규칙을 통해 변환된 S-BPM은 본 기법에서 핵심이 되는 워크플로우의 경로 정보를 그대로 유지하며BPMN의 다양한 부가 정보 표현은 손실되지만 여전히 BPMN의 규칙을 따른다.

3.3 S-BPM기반 단순 서비스 경로 그래프 생성(단계 3)

단계 2에서 만들어진 S-BPM은 다양한 정보를 축소하여 간결해졌으나 방향성 그래프와 같은 일반적인 자료구조와는 다른 형태를 가지고 있어 최소 개수로 구성되는 최단거리 경로정보를 뽑아내기 위해서는 특화된 알고리즘이 필요하다. 이것은 기존의 많은 알고리즘들을 활용할 수 없어 전용의 알고리즘을 추가로 연구하여야 하는 문제점을 야기시킨다. 따라서 기존의 방향성 그래프 알고리즘들을 본 기법에서도 사용할 수 있도록 S-BPM을 방향성 그래프로 변환한다. 방향성 그래프는 S-BPM의 경로 정보는 유지하면서 알고리즘의 입력이 될 수 있도록 노드와 간선만으로 이루어진 형태로 생성되며 이러한 방향성 그래프를 단순 서비스 경로 그래프라 칭한다. 단순 서비스 경로 그래프를 생성하기 위해서는 S-BPM의 요소들 각각에 생성 규칙을 적용한다. 이 규칙은 생성 가능한 경로의 개수와 그 경로에서 사용되는 Activity의 구성은 완전히 유지하면서 방향성 그래프로 만족하도록 함을 원칙으로 한다. 다음 소절을 통해서 먼저 단순 서비스 경로 그래프의 정의를 기술하고, S-BPM을 기반으로 단순 서비스 경로 그래프를 생성하는 규칙을 설명한다.

3.3.1 단순 서비스 경로 그래프의 정의

단순 서비스 경로 그래프는 S-BPM에서 생성 가능한 모든 경로의 개수를 유지하고, 그 경로에서 사용되는 Activity의 구성 역시 동일하도록 만들어진 방향성 그래프이다. 그러나 S-BPM의 풍부한 표현력 때문에 방향성 그래프에선 표현이 어려운 정보가 존재하므로 일부 요소들은 유사한 표현으로 대체된다. 이러한 요소로는 AND Gateway로 인해 동기화되어 동시에 동작하는 2 개 이상의 부분 프로세스가 있다. 이러한 동기화 된 경로들을 단순 서비스 경로 그래프에서는 하나의 경로에서 순차적으로 동작하는 것으로 바꾸어 표현하지만 전체 경로에서 동작하는 Activity는 변화하지 않는다. 다음은 단순 서비스 경로 그래프의 정의이다.

[정의 2 단순 서비스 경로 그래프] 단순 서비스 경로 그래프는 서비스 시스템의 워크플로우에서 수행 되는 행위, 진행되는 순서와 방향 정보를 표현하는 방향성 그래프이다. 그 구성은 다음과 같다.

가) Activity: 서비스의 워크플로우에서 수행되는 단일 작업, 방향성 그래프에서 하나의 노드

나) Sequence Flow: 워크플로우에서 하나의 활동에서 다음 활동으로 진행되는 순서, 방향과 연결 정보를 포함하는 하나의 간선

3.3.2 S-BPM 기반 단순 서비스 경로 그래프 생성 규칙

이전 단계에서 S-BPM으로 변환되면서 남은 BPMN 요소는 Event, Gateway, Sequence Flow의 3 가지이다. 각 요소마다 제안 규칙을 적용하여 단순 서비스 경로 그래프를 생성한다. 가장 먼저 규칙을 적용해야 할 요소는 Gateway로 그래프의 간선과 경로를 가장 크게 변경 시킬 수 있는 요소이다. 다음 <표 2>는 S-BPM의 BPMN 요소 중 Gateway에 대하여 선조건과 후조건을 표기하고, 그에 따른 규칙을 정의한 것이다. 선조건은 해당 BPMN 요소의 직전 프로세스 BPM 상태를 나타내며, 후조건은 해당 BPMN 요소 직후의 BPM의 상태를 나타내고 이에 따라 생성 규칙이 결정된다. 활동이란 BPMN에서의 Activity이며, 경로란 활동과 Sequence Flow로 이루어진 워크플로우의 부분 집합이다. 또한 Gateway의 전, 후에 파생되는 경로들은 2개 이상이 될 수 있으며, 경로가 늘어나더라도 <표 2>의 규칙을 반복 적용하여 같은 결과를 만들 수 있다.

XOR-Split은 선조건으로 활동 A가 있고, Gateway를 통과하여 후조건인 경로 B나 경로C중에 하나의 경로만 선택하여 진행되는 사례로써 전체 경로 개수를 후조건 경로의 개수만큼 늘어나게 하는 요소이다. 따라서 방향성 그래프에서도 이러한 특성을 그대로 반영하기 위해, BPM에서 XOR Gateway를 제외하고 활동 A에서 경로 B와 경로 C로 이어주는 2개의 간선을 연결함으로써 방향성 그래프에서 경로 정보를 유지하면서 동일한 정보를 표현할 수 있다.

XOR-Join은 선조건으로 경로 A와 경로 B가 Gateway에 연결되고 이중 하나의 처리 결과만 선택하여 후조건인 활동 C에게 넘겨주는 사례로 XOR-Split과 유사한 모습을 보인다. 따라서 방향성 그래프 생성시 Gateway는 제외하고, 경로 A와 경로 B로부터 시작되는 2개의 간선이 활동 C를 연

〈표 2〉 S-BPM의 Gateway에 대한 단순 서비스 경로 그래프 생성 규칙

BPMN 요소	선조건	후조건	규칙	비고
OR-Split ◈	활동 A	경로 B 경로 C	A →[B ⊕ C]	우선순위 1 […]: Sequence Set →: Next ⊕: XOR
XOR-Join ◈	경로 A 경로 B	활동 C	[A ⊕ B] →C	우선순위 2 AND-Split 선행 시 AND-Split 규칙 생략
AND-Split ◈	활동 A	경로 B 경로 C	[A→B→C]	우선순위 3
AND-Join ◈	경로 A 경로 B	활동C	[A→B→C]	우선순위 4

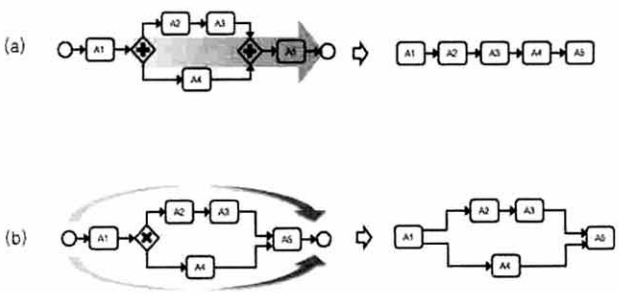
결하는 형태로 생성한다. 만약 경로 A와 경로 B의 분기점이 AND-Split일 경우 XOR-Split 변경 규칙을 먼저 적용한 후 AND-Split을 생략하는 것으로 규칙 적용이 완료된다.

AND-Split은 선조건으로 활동 A가 연결되고 Gateway를 통과하여 후조건에 경로 B와 경로 C가 동기화 동작하는 사례로, 방향성 그래프에서 동기화 정보는 표기할 수 없으므로 수정해야 한다. BPM에서는 두 개의 경로가 동기화 진행되어도 결국 하나의 시나리오 내에서 진행되므로 방향성 그래프에서는 이를 단일 경로로 통합하여 생성한다. 해당 과정을 통해서 경로 B와 경로 C가 동시에 수행되던 표기법이 경로 B가 수행된 후에 경로 C가 수행되는 모습으로 변하지만 경로 전체에서 수행되는 활동은 동일하며 순서의 정보만 변형된다. AND Gateway의 특성상 모든 활동이 실행되어야 하기 때문에 순서가 변형된 활동들이라 할지라도 해당 활동이 모두 수행된다면 서비스 시나리오 추출에는 아무런 지장이 없다. 따라서 경로 B와 경로 C의 우선순위는 어느 쪽을 먼저 선택 하더라도 관계가 없으며 표현의 편의에 따라서 선택한다.

AND-Join의 경우에는 경로 A와 경로 B가 동기화되어 수행결과가 AND-Join에서 합쳐지게 되고 다음 활동 C가 진행되는 사례이고 이것은 AND-Split과 유사한 모습을 보인다. 따라서 같은 이유로 방향성 그래프에서는 경로 A가 수행되고 경로 B가 수행된 후에 활동 C로 넘어가는 형태로 변경된다. 보통의 경우에는 AND-Join은 AND-Split 뒤에 사용되게 되므로 AND-Split에 위의 규칙을 먼저 적용한 경우 AND-Join은 변경할 것이 없이 생략된다.

각 Gateway의 특성상 변형 규칙의 적용 순서는 XOR-Split, XOR-Join, AND-Split 그리고 마지막으로 AND-Join이 되는 것이 중복처리를 줄이고 규칙간의 충돌이 발생하는 것을 방지할 수 있어 권장된다.

(그림 6)은 위의 규칙을 적용한 예제로 AND-Split과 AND-Join이 함께 쓰인 BPM을 (a)에서 보여주고, XOR-Split이 사용된 BPM을 (b)에서 보여준다. (a)에서는 동기화 프로세스가 2개 있으나 동시에 실행되어 1개의 시나리오가 되기 때문에 단순 서비스 경로 그래프는 1개의 경로로 생성된다. (b)는 XOR-Split 때문에 (A1,A2, A3,A5)와 (A1,A4,A5)로 진행되는 2개의 시나리오가 존재하며 이를 표현하기 위해



(그림 6) 단순 서비스 경로 그래프로 변경 예제

단순 서비스 경로 그래프는 분기 경로를 가지는 형태로 생성된다. 예제를 보면 S-BPM과 단순 서비스 경로 그래프는 형태의 변환은 있으나 시나리오의 개수에는 변화가 없음을 알 수 있다.

Sequence Flow는 Gateway를 변경하면서 만들어진 새로운 Sequence Flow를 포함하여 모두 방향성 간선으로 변경한다. 호칭과 표기법만 변경될 뿐 S-BPM과 동일한 정보를 표현한다.

Event는 방향성 그래프에서 나타낼 수 없는 요소로써 규칙에 따른 표현 변환이 필요하다. 다음 <표 3>은 단순 서비스 경로 그래프로 변경하기 위한 Event에 해당하는 요소들의 규칙으로, 가장 마지막에 적용하는 것이 효율적이다.

Start Event는 BPM에서 시작점이 되는 이벤트로써 사용자의 직접적인 개입이 있는 입력으로 알고리즘에서는 소스 노드라고 표현되며 Start Event 직후의 활동을 시작지점으로 사용할 수 있으므로 생략한다.

Intermediate Event는 BPM에서 Start와 End사이에만 위치할 수 있는 이벤트로써 여러 조건에 따라서 프로세스 진행을 지연시키거나 진행시킨다. 두 가지 사례로 사용되어 Sequence Flow 중간에 위치하여 이벤트 조건이 만족될 때까지 프로세스를 지연시키는 경우와 Activity에 사용되어 조건이 만족되면 해당되는 경로로 진행되는 경우가 있다. 지연 상태는 프로세스를 종료한 것이 아니므로 본 기법에서는 언젠가 진행될 것으로 가정하고, 경로의 개수에는 관여하지 않기 때문에 생략한다. Activity에 사용된 경우에는 일종의 분기점으로 볼 수 있으며 조건이 만족되면 해당 경로가 진행되기 때문에 Event는 생략하되 다음 활동과 연결하는 간

〈표 3〉 S-BPM의 Event에 대한 단순 서비스 경로 그래프 생성 규칙

S-BPM Event	생성 규칙	참고
Start	1. Start Event 생략 2. Start Event 다음 활동이 단순 서비스 경로 그래프에서의 시작 노드가 됨	
Intermediate	1. Sequence Flow 중간에 위치 A. Intermediate Event 생략 B. Intermediate Event 앞의 활동과 뒤의 활동을 Sequence Flow 로 연결 2. Activity에 포함 A. Intermediate Event 생략 B. Intermediate Event 를 포함하는 Activity와 다음 활동과의 연결 간선은 유지	
End	1. End Event를 노드로 변경하며 종료 지점임을 표기 2. 각 End 노드마다 Start Event 부터 End Event까지 이어지는 모든 경로로만 이루어진 그래프를 생성	• End Event 개수만큼의 단순 서비스 경로 그래프가 생성

선을 유지하여 경로 정보를 보존한다.

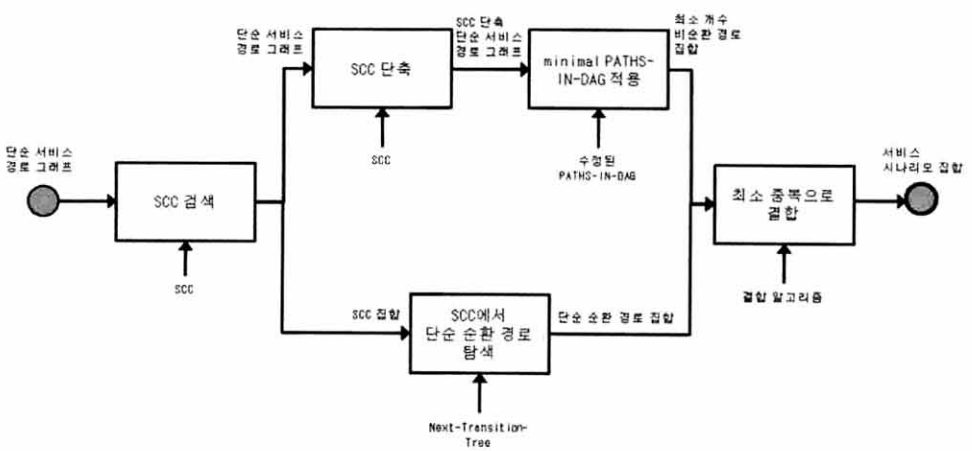
End Event는 BPM에서 종료지점으로 사용되는 이벤트로써 사용자에게 결과물을 전달하며 알고리즘에서는 말단 노드로 표기된다. 하나의 BPM에는 하나의 Start Event만 존재하여야 함과 다르게 End Event는 여러 개가 존재할 수 있다. 이러한 종료 지점을 방향성 그래프에서도 기억할 수 있도록 모든 End Event를 노드로 변경하면서 종료 지점임을 표기한다. 또한 그래프의 단순화를 위하여 각 종료 노드마다 시작지점부터 종료지점까지 거칠 수 있는 모든 경로 집합만을 가지는 그래프를 생성한다. 만약 3개의 End Event가 있다면 최종 단순 경로 그래프는 3개가 생성되며 이를 통하여 각 종료지점마다 불필요한 정보를 생략하여 알고리즘의 효율을 높일 수 있다.

본 단계를 통해 초기 BPM에서 워크플로우의 경로 정보는 그대로 보존하면서 기존 알고리즘들을 적용할 수 있는 방향성 그래프인 단순 서비스 경로 그래프가 생성된다.

3.4 서비스 시나리오 생성(단계 4)

앞의 단계에서 만들어진 단순 서비스 경로 그래프는 비즈니스 프로세스에 따라서 복잡한 경로를 가질 수 있다. 특히 순환 경로와 경로 분기의 증가에 따라서 생성 가능한 경로의 숫자가 기하급수적으로 늘어나게 된다. 이러한 경로들에서 최소개수로 구성되는 최단거리 경로들만 뽑아내어 테스트 케이스의 효율을 높이기 위하여 기존의 알고리즘들을 활용한다. 단순 서비스 경로 그래프에는 방향성 그래프에 적용이 가능한 많은 기존 알고리즘 중에 목적에 따라서 원하는 것을 선택하여 사용할 수 있다. 본 기법은 앞서 2.3절에서 언급한 DAG와 Next-Transition-Tree를 사용한 전체 경로 검색 알고리즘을 최소 중복 경로 검색 알고리즘으로 수정하여 적용한다. DAG와 Next-Transition-Tree는 함께 사용되어 순환경로가 존재하는 복잡한 그래프에서 단순한 비순환 경로를 쉽게 찾아 낼 수 있어 기존 연구들[16,24]에서도 사용되었다.

다음 (그림 7)은 실제 적용할 서비스의 전체 시나리오 검색 알고리즘의 과정을 간략하게 도식화한 것이다. 역시 IDEF0 표기법을 사용하여 왼쪽 화살표가 입력, 오른쪽 화살표는 출력, 하단 화살표는 관련 기법을 표현한다. 진행 과정 중에 앞서 기술한 DAG와 SCC, Next-Transition-Tree가 사용되며 특히 순환경로와 같은 복잡한 경로를 단순화 하는 것이 주목적이다.



(그림 7) 서비스의 전체 시나리오 검색 과정

이 알고리즘의 목표는 그래프에서 생성될 수 있는 모든 간선을 최소한 한번 이상 탐색할 수 있는 완전성과 이런 점을 만족하면서 중복을 최소화하여 경로 집합을 생성하는 것이다. 알고리즘은 크게 4단계로 나눌 수 있다. 먼저 입력인 단순 서비스 경로 그래프에서 SCC를 찾는다. 다음에 SCC를 단축하고 시나리오 검색 알고리즘을 적용하여 최소 개수 비순환 경로 집합을 만든다. 동시에 SCC에서 더 이상 작은 순환 경로를 포함하고 있지 않은 순환 경로를 의미하는 단순 순환 경로 모두 찾는다. 마지막으로 위의 두 집합을 합침으로 서비스 시나리오 집합을 완성한다.

다음은 기존의 상태전이그래프 경로 검색 알고리즘[16]을 본 기법의 목적에 맞도록 수정한 서비스의 전체 시나리오 검색 알고리즘이다.

본 알고리즘은 입력으로 앞서 생성된 단순 서비스 경로 그래프를 사용하고, 출력으로 서비스 시나리오 집합을 생성하며 서비스 시나리오 집합은 본 논문의 목표가 되는 최소 개수로 구성되는 최단거리의 경로를 만족한다. 이러한 결과를 위해서 입력인 단순 서비스 경로 그래프 G에서 SCC를 검색하고 SCC에서 Next-Transition-Tree를 생성한다. 다시 Next-Transition-Tree를 통해서 단순 순환 경로를 찾아내어 집합 C를 만든다. 다음으로 단순 서비스 경로 그래프 G에서 찾아진 SCC를 단축하여 노드로 치환하고 minimal PATHS-IN-DAG를 적용하여 집합 P'를 얻는다. minimal PATHS-IN-DAG는 기존의 PATHS-IN-DAG가 전체 경로를 검색하는 것을 최소개수로 구성되는 최단거리경로만 검색하도록 수정한 알고리즘이다. 다음은 수정한 최단거리 경로 검색 알고리즘이다. 집합 P'의 각 경로 p' 마다 SCC가 단축된 노드를 다시 SCC에서 단축된 노드의 앞 뒤를 연결할 수 있는 최단거리 경로 scp를 탐색하여 치환하여 경로 p를 생성하고 이렇게 치환된 경로 p를 모두 모아 집합 P를 만든다. 마지막으로 집합 C와 집합 P를 병합하기 위한 과정을 거치는데 집합 P의 경로와 집합 C의 경로의 중복 사용을 최소화는 것이 중점이 된다. 병합 시에는 경로 p에서 scp의 바로 뒤에 scp로 시작하는 경로c를 추가함으로써 병합 경로 s가 생성되며 모든 c와 p가 한번 이상 사용될 때까지 위의 동작을 반복하여 경로 s의 집합인 S를 얻고 이를 서비스 시나리오 집합이라 한다.

본 알고리즘은 이전 단계에서 생성된 모든 단순 서비스 경로 그래프에 적용하고 각각 생성된 모든 서비스 시나리오를 수집하여 전체 서비스 시나리오 집합을 얻음으로써 본 단계를 종료한다. 본 단계를 통해 테스트 목표로 지정된 하나의 서비스 시스템에서 생성될 수 있는 최소 개수의 최단거리 경로가 탐색되며 경로들은 하나의 서비스에서 생성될

서비스의 전체 시나리오 검색 알고리즘

입력. 단순 서비스 경로 그래프 집합 G
출력. 전체-범위를 충족하는 최소한의 서비스 시나리오 집합 S

```
1.  find all SCC in G;
2.  For each SCC
3.      For each node x in SCC
4.          if 노드 x 가 DAG와 연결된 시작노드 then
5.              SCC에 대한 Next-Transition-Tree 생성
6.              C ←search simple cyclic
                paths(Next-Transition-Tree)
7.  G' ←replace all SCC as 노드Si /* G' 는 DAG , i는 각SCC의
    번호*/
8.  P' ← minimal PATHS-IN-DAG(G' ) /* P' 는 최소 개수 비순환
    소스-말단 경로 집합 */
9.  For P' 의 각 경로 p'
10.     For p' 의 각 노드 Si
11.         scp←search(SCC에서 Si 전후의 노드를 연결할 수
            있는 shortest cyclic path)
12.         p←replace (Si, scp)          /* Si 를 scp로 치환
            */
13. P←collect all p                      /* P는 비순환 경로
    집합*/
14. While 모든 p가 사용중 & 모든c 가 사용중{/* P와 C의 경로가
    모두 최소1번 이상 사용*/
15. If pj has scp then                                  /* j
    는 1부터 p의 총 개수 까지*/
16.         While pj has scp   {          /* i 는 1부터 c의
            총 개수 까지*/
17.             If (scp로 시작하는 ci 가 미사용) or (scp로
                시작하는 모든 c 가 사용) then
18.                 replace(scp, scp로 시작하는 ci) /* scp를
                    scp로 시작하는 ci 로 치환 */
19.                 ci 를 사용 표시
20.                 pj 를 사용 표시
21.                 집합S에 pj 추가
22.                 j와 i를 1증가
23.             else
24.                 i를 1증가
25.         }
26.     else
27.         pj 를 사용 표시
28.         j 를 1 증가
29.         집합 S에 pj 추가
30. }
31. return(S)
```

Minimal PATHS-IN-DAG 알고리즘

입력. 하나의 소스 s와 여러 개의 말단 간의 DAG인 G'
출력. G' 에서 모든 소스-말단 경로
참고. p(v_i) 는 G' 에서 모든 v_i-말단 경로를 표현

```
1.  TopologicalSort(G' ); /* G'  = {v0,v1, …, vn-1} */
2.  For i = 0 to n-1
3.      노드 vi 로 향하는 간선의 시작점인 노드를 w1, …, wr라
        하면;
4.      모든 (wj, vi)을 미사용 표시
5.  While 모든 (wj, vi)가 사용 상태가 아님 { /* 모든 간선을 최소1번
    사용 */
6.      i=n-1
7.      While vi ' v0 {
8.          j ← 1 /* j는 1부터 r 까지 */
9.          While vi ' wj  {
10.             if (wj, vi) = 미사용 or j = r then
11.                 p(vi) ← (wj, vi) + p(vi)
12.                 (wj, vi)에 사용 표시
13.                 vi ←wj
14.             else
15.                 j를 1증가 /* 다음 노드로 */
16.             }
17.     }
18.     p(t) ← p(t) + p(v0)
19. }
20. return p(t)
```

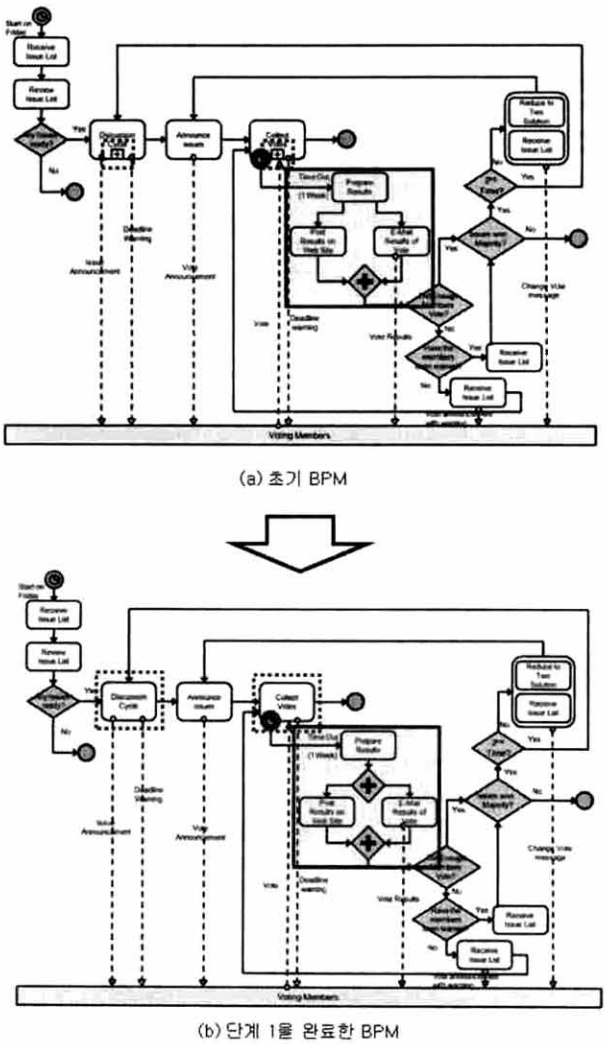
수 있는 많은 경로 중에서 최소한의 실행으로 모든 간선을 테스트할 수 있는 경로들이다. 여기서 하나의 간선은 하나의 활동에서 다른 활동들로 전송하는 메시지의 경로를 의미하고 이렇게 탐색된 경로들은 하나의 서비스 시스템에서 나타나는 최소개수로 구성되는 최단거리 시나리오 집합으로써 이를 서비스 시나리오 집합이라고 정의한다. 시나리오란 완성된 소프트웨어가 외형적으로 어떤 기능을 가져야 하는가를 입력 조건에 따른 반응으로 나타낸 것[20]으로, 본 논문에서는 서비스 시스템을 완성된 소프트웨어로 본다. 시나리오는 그 자체로 테스트 케이스로 사용되기도 하고 형식적인 표현으로 변환시켜서 테스트 케이스를 만들기도 한다[21].

3.5 테스트 케이스 생성(단계 5)

앞서 언급한 바에 따르면 단계 4까지 완료하는 것으로 테스트 케이스가 생성되었다고 볼 수도 있다. 그러나 서비스 시나리오만으로는 본 논문의 목표인 메시지 이동에 대한 표현이 부족하기 때문에 본 단계에서 서비스 시나리오로부터 메시지 이동 경로를 표현하는 방법을 서술한다. 본 단계는 기존의 기법을 활용하여 서비스 시나리오마다 UML의 순차 다이어그램(Sequence diagram)의 표기법을 사용하여 서비스 시스템의 순차 다이어그램을 작성한다. 순차 다이어그램의 각 객체들은 본 기법에서는 컴포넌트로 표현되며, 각 객체 간에 주고 받는 이벤트 혹은 메시지는 실제 서비스 시스템 내에서 주고받는 메시지 흐름이 된다. 이러한 표현으로 각 서비스 시나리오마다 생성한 순차 다이어그램을 서비스 순차 다이어그램이라 부르며, 각 서비스 순차 다이어그램의 메시지들을 입력부터 종료 시점까지 나열하여 하나의 집합을 얻는다. 하나의 집합이 하나의 테스트 케이스를 형성한다. 따라서 모든 서비스 시나리오의 서비스 순차 다이어그램을 작성하고 메시지 집합을 추출함으로써 목표 서비스 시스템의 전체 서비스 테스트 케이스를 얻을 수 있다.

4. 적용 사례

본 장에서는 제안 기법의 적용 사례를 보여 실제 적용 가능성을 보이고, 제안 기법의 이해를 돕는다. 적용 사례에서 사용된 BPM은 BPMI가 2004년도에 만든 BPMN 1.0버전 문서에 포함되어 있는 예제 중 하나로 BPMN을 사용하여 만들어 졌다. (그림 8)의 (a)는 사용된 예제 BPM이며 e-mail 투표를 통해 논쟁점을 해결하기 위한 비즈니스 프로세스를 표현한 것이다. 이 프로세스는 작지만 상당히 복잡하며 BPMN의 특징을 나타내기 위한 예제로 기존 연구[22]에서도 사용되었다. 무한 반복 가능한 순환 경로와 같은 비즈니스 프로세스를 가지고 있는 등 전형적인 프로세스는 아니지만 BPMN이 복잡하면서도 일반적이지 않은 비즈니스 프로세스까지 다룰 수 있음에도 불구하고 쉽게 이해할 수 있음을 보여주는 예시이다[23]. 따라서 본 기법의 적용을 보이기에도 적절한 사례라 판단되며 다음을 통해 제안 기법의 각 단계별 적용 사례를 보인다.



(a) 초기 BPM

(b) 단계 1을 완료한 BPM

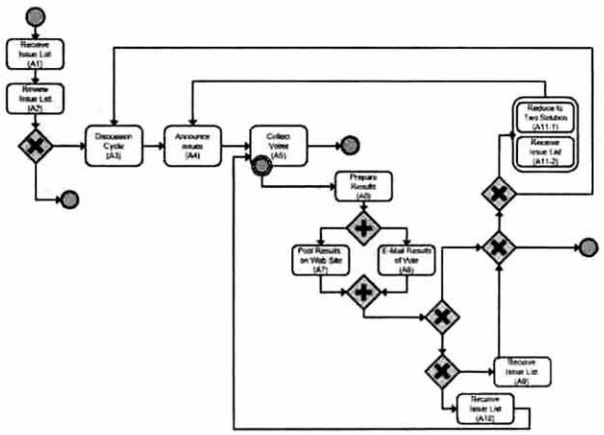
(그림 8) 적용 사례에 사용된 e-mail 투표 비즈니스 프로세스의 BPM

4.1 서비스 범위 설정 및 BPM 모델화 적용

SOA 개발에서 모델링 단계에 만들어진 초기 BPM인 (그림8)의 (a)는 BPMN이 가지고 있는 요소들을 상당수 표현하고 있는 복잡한 BPM이다. 해당 BPM은 접힌 프로세스를 가지고 있으며 혼란이 올 수 있는 경로 분기점을 포함하고 있다. (그림 8)의 (a)에서 점선으로 표시하고 있는 부분이 접힌 프로세스이며 단계 1에서 서브 프로세스를 테스트 범위에 넣지 않기로 결정하고 일반 Activity로 단순화 시킨 것이 (b)에서 점선으로 표시된 부분이다. 또한 (그림 8)의 (a)에서 실선의 원으로 표시된 부분은 경로 분기점과 AND-Join이 함께 사용된 부분으로 BPM을 읽는 사람에 따라서 혼란을 가져올 수 있는 부분으로 명확히 하기 위해 (b)의 실선의 원 부분과 같이 AND-Split을 추가하여 명확한 표기로 수정한다.

4.2 BPM-to-S-BPM 모델 변환 적용

초기 BPM은 경로 정보만 찾아내기엔 대단히 복잡하므로



(그림 9) 초기 BPM이 변환된 S-BPM

S-BPM으로 변경시킨다. (그림 9)는 단계 2를 적용하여 초기 BPM이 변환된 S-BPM이다.

S-BPM은 (그림 8)의 (b)에서 보인 초기 BPM보다는 단순해진 모습을 보인다. Message Flow와 Pool과 같은 요소가 생략 되었고, Event들도 세부 내용이 생략된 None Event로 표기 되었으며 일반 Gateway는 XOR Gateway로 변경되었다. 그러나 워크플로우 경로는 초기와 완전히 동일한 표기를 보여준다. 이것은 예제 BPM이 다양한 요소를 포함하고 있으나 AND와 일반 Gateway만 사용하고 있어 단순한 Gateway 대체와 일부 요소 생략만으로 S-BPM으로의 변경 과정이 완료 되기 때문이다.

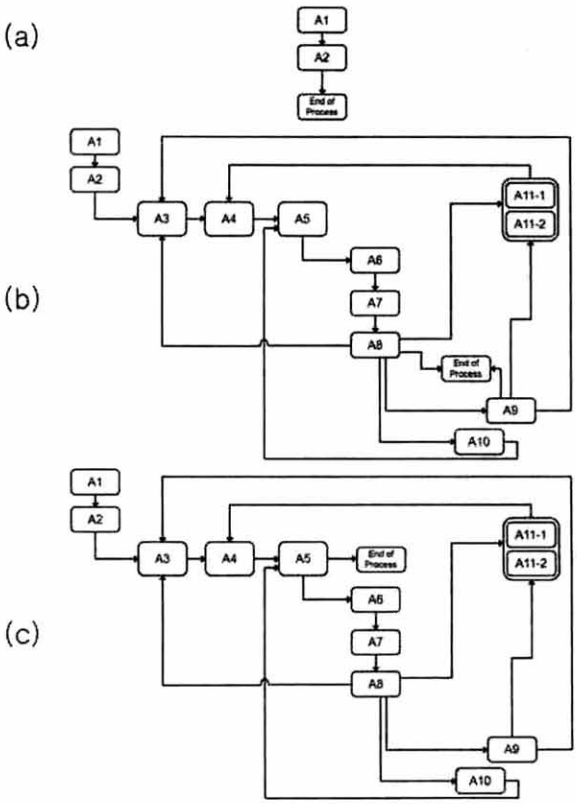
4.3 S-BPM기반 단순 서비스 경로 그래프 생성(단계 3)

S-BPM을 기반으로 알고리즘이 적용될 수 있는 단순 서비스 경로 그래프를 생성해야 한다. XOR Gateway가 단순한 분기점이 되고, AND Gateway 부분이 단일 경로로 변경된다. End Event마다 하나의 단순 서비스 경로 그래프가 나오기 때문에 시작점부터 각 End Event까지 경로로만 이루어진 총 3개의 그래프가 생성되며 (그림 10)에서 이를 보여준다.

Activity의 개수가 많기 때문에 편의를 위해 단순한 이름으로 표기하였으며 (그림 9)의 각 Activity의 괄호 속 이름을 참조한다.

4.4 서비스 시나리오 생성 적용

위의 그래프들에 알고리즘을 적용하여 생성되는 서비스 시나리오는 총 5개로 다음의 <표 3>을 통해 보인다. 위에서부터 (a)는 A 그래프, (b)는 B 그래프, (c)는 C 그래프로 칭



(그림 10) 생성된 3개의 단순 서비스 경로 그래프

한다.

이러한 과정을 통하여 생성된 서비스 시나리오는 활동의 순서 집합으로 표기되며, 이 순서대로 동작하는 경로를 모두 탐색함으로써 해당 서비스 시스템에서 발생할 수 있는 활동과 활동 사이에 존재하는 메시지 경로를 최소 1회 이상 수행하여 볼 수 있다. 서비스 시나리오가 생성된 후에 서비스 순차 다이어그램을 작성하여 각 컴포넌트 사이의 메시지 이동을 추출하여 메시지의 순차 집합을 생성함으로써 테스트 케이스기 완성된다. 순차 다이어그램을 작성하는 것은 널리 알려진 기법이며 기존 연구[4]에서도 사용한 것을 재정의한 것이므로 해당 적용을 보이는 것은 생략한다.

5. 평 가

본 장에서는 제시한 테스트 케이스 생성 기법에 대해 정량적 평가와 정성적 평가를 통해서 그 효율성을 평가한다. 먼저 정량적 평가에서 기법을 적용하지 않은 일반 BPM을

〈표 3〉 생성된 서비스 시나리오 집합

그래프	생성된 서비스 시나리오
A 그래프	(A1, A2, End)
B 그래프	(A1, A2, A3,A4, A5, A6, A7,A 8, A3, A4, A5, A6, A7, A8, A11, A4, A5, A6, A7, A8, A10, A5, A6, A7, A8 End), (A1, A2, A3,A4, A5, A6, A7,A 8, A9, A3, A4, A5, A6, A7, A8, A9, A11, A4, A5, A6, A7, A8, A9 End)
C 그래프	(A1, A2, A3,A4, A5, A6, A7,A 8, A3, A4, A5, A6, A7, A8, A11, A4, A5, A6, A7, A8, A10, A5, End), (A1, A2, A3,A4, A5, A6, A7,A 8, A9, A3, A4, A5, A6, A7, A8, A9, A11, A4, A5, End)

통해 생성 가능한 시나리오의 개수와 본 기법을 적용하여 생성한 시나리오의 개수를 비교함으로써 비용의 효율성을 보인다. 다음 정성적인 평가에서는 비교 평가 기준을 작성하여 기존의 연구들과 비교 평가한다.

5.1 생성 가능 시나리오 수의 정량적 평가

본 기법은 최대 효율의 커버리지(Coverage)를 목표로 한다. 이것은 서비스 시스템에서 존재하는 모든 메시지 경로를 한번 이상 테스트할 수 있는 최소한의 경로 집합을 탐색하는 것을 말한다. 이러한 목표가 실현되었는지 확인하기 위하여 본 평가에서는 기법을 적용하지 않은 일반적인 BPM에서 생성될 수 있는 서비스 시나리오의 개수와 동일 BPM에 본 기법을 실제 적용하여 생성된 서비스 시나리오 개수를 비교한다. 하나의 서비스 시나리오는 하나의 테스트 케이스를 생성하므로 서비스 시나리오의 개수 비교만으로도 테스트 케이스 개수를 비교하는 것과 동일한 결과를 얻을 수 있다. 이러한 평가를 통하여 테스트 케이스 생성의 효율이 증가하였음을 확인하고 따라서 테스트 비용이 감소함을 증명한다.

비교를 위한 기준은 BPM 경로 복잡도라 명하고 BPM 경로 복잡도가 증가함에 따른 생성된 시나리오 개수를 비교한다. BPM 경로 복잡도란 BPM이 생성 가능한 서비스 시나리오의 개수가 증가될 수 있는 요소들을 얼마나 많이 포함하고 있는가를 표현하는 척도로써 BPM의 요소 중 Gateway와 순환경로가 가장 큰 영향을 미친다. Gateway는 BPM에 다양한 분기점을 만들어 서비스 시나리오의 개수를 증가시키며, Gateway중 AND-Split과 XOR-Split, 경로 분기점이 해당된다. Split은 워크플로우의 분기점을 만들어내 시나리오의 개수를 증가시키지만, Join은 워크플로우의 분기점을 줄여 서비스 시나리오의 개수를 늘이지 않으며, 보통 Split 이후에 생성되므로 BPM 경로 복잡도에서는 제외 시킨다. 경로 분기점은 일반적으로 Split을 사용하는 것이 옳지만 사용되었을 경우 Gateway로 간주하며 Split과 같이 복잡도를 증가 시킨다. BPM 경로 복잡도는 위의 요소에서 파생되는 기본 2개의 간선에 1을 증가시키고 추가 간선 마다 1씩 더 증가한다. 순환경로는 워크플로우에서 이전 프로세스로 돌아가도록 하는 회귀간선으로 인하여 발생하며 워크플로우의 실행 횟수를 늘여 시나리오의 개수를 증가시키므로 회귀간선마다 BPM 경로 복잡도를 1씩 증가시킨다. 따라서 BPM 경로 복잡도는 AND-Split과 XOR-Split, 경로 분기점, 그리고 회귀간선의 합으로 정의한다. 다음은 BPM 경로 복잡도를 계산하는 공식을 정리한 것이다.

*BPM*경로 복잡도

$$C = \sum_{n=1}^{n}(\text{AND}-\text{Split 분기경로수}) - n + \sum_{m=1}^{m}(\text{XOR}-\text{Split 분기경로수}) - m + \sum_{l=1}^{l}(\text{분기경로수}) - l + \sum_{x=1}^{x} x$$

n: AND-Split Gateway의 개수
m: XOR-Split Gateway의 개수
l: 경로 분기점의 개수
x: 회귀간선의 개수

일반 BPM에서 생성 가능한 서비스 시나리오 개수를 산출할 때 동일한 순환경로는 하나의 시나리오에서 최대 1 회만 거치는 것을 기준으로 한다. 만약 동일 순환 경로를 2회 이상 거칠 수 있도록 할 경우 무한 루프에 빠지거나 과도한 순환 반복을 하는 경우가 발생하며 이때 시나리오의 개수는 기하급수적으로 증가한다. 표본으로 사용된 BPM은 BPMN관련 사이트와 책자에서 추출한 것으로 모든 BPM을 대표할 수 있는 평균적인 BPM은 아니지만 6개의 대표 표본을 사용함으로써 어떠한 BPM에도 제안 기법이 적용되고 효율성이 증가함을 보인다. 6개의 BPM은 다시 두 가지 집단으로 분류 하였다. 비교 집단 1은 BPM 경로 복잡도가 1, 2, 4, 8로 2의 배수로 증가되며 BPM 경로 복잡도 증가에 따른 제안 기법의 효율을 평가한다. BPM 경로 복잡도 1의 BPM은 비용에 따라 승인 활동이 변하는 프로세스이고, BPM 경로 복잡도 2의 BPM은 주식 구매 프로세스이다. BPM 경로 복잡도 4의 BPM은 적용사례에 사용한 BPM의 서브 프로세스 중 하나인 Discussion Cycle이다. BPM 경로 복잡도 8의 BPM은 자원 배치 프로세스로써 3개의 End Event를 가지고 있는 특징이 있다.

비교 집단 2는 8, 9, 10의 높은 BPM 경로 복잡도와 순환경로까지 포함하는 특징이 있어 복잡한 BPM에 대한 제안 기법의 효율을 평가한다. BPM 경로 복잡도 8의 BPM은 비교집단 1에서 사용된 것과 동일하며, . BPM 경로 복잡도 9의 BPM은 고객 지원 프로세스로 약간의 순환 경로를 가진다. 특히 BPM 경로 복잡도 10의 BPM은 적용사례에서 사용한 e-mail 투표에 관한 BPM으로 대단히 복잡한 순환경로를 가지고 있다.

평가 결과에서 감소 비율이란 일반 BPM의 서비스 시나리오 개수에 비하여 제안 기법의 서비스 시나리오 개수가 얼마나 적어지는 지를 나타내는 지표로써 감소 비율 R은 다음 공식을 통하여 계산한다.

$$\text{감소비율} R = \frac{Sn - Pn}{Sn} * 100$$

Sn: 일반 BPM으로 도출되는 서비스 시나리오 개수
Pn: 제안기법으로 도출되는 서비스 시나리오 개수

다음 <표 4>는 비교 집단 1에 기법을 적용하지 않은 경우 생성 가능한 서비스 시나리오의 개수와 제안 기법으로 인해 생성되는 서비스 시나리오의 개수를 비교한 평과 결과이다. BPM 경로 복잡도와 회귀간선의 수, 그리고 순환경로에 관련된 간선의 수는 각 BPM에 따라서 고유의 수치이며 일반 BPM 시나리오 개수는 BPM에 제안 기법을 적용하지 않은 경우에 생성 가능한 시나리오 개수, 제안기법 시나리오 개수는 제안기법을 적용하여 생성된 경우의 시나리오 개수를 의미한다.

비교집합 1의 결과를 볼 때 특히 복잡도가 낮으면서 순환경로가 없는 경우 제안 기법을 적용하여도 효율성 증가가 미약하다. 그러나 복잡도가 증가할수록 제안 기법으로 인한

〈표 4〉 비교 집단 1의 제안 기법 적용 결과 평가

BPM 복잡도	회귀간선수	순환경로 관련 간선수	일반 BPM 시나리오 개수	제안기법 시나리오 개수	시나리오 개수 감소 비율
1	0	0	2	2	0.00%
2	0	0	3	3	0.00%
4	0	0	4	2	50.00%
8	3	4	8	3	62.50%

〈표 5〉 비교 집단 2의 제안 기법 적용 결과 평가

BPM 복잡도	회귀간선수	순환경로 관련 간선수	일반 BPM 시나리오 개수	제안기법 시나리오 개수	시나리오 개수 감소 비율
8	3	4	8	3	62.50%
9	2	4	20	4	80.00%
10	4	9	919	5	99.31%

시나리오 개수 감소비율이 증가하기 시작한다. 이러한 결과는 BPM이 복잡할수록 제안 기법으로 인한 효율이 증가함을 알 수 있다.

다음 <표 5>는 비교 집단 2에 기법을 적용하지 않은 경우, 생성 가능한 서비스 시나리오의 개수와 제안 기법으로 인해 생성되는 서비스 시나리오의 개수를 비교한 평과 결과이다. 특히 비교 집단2는 복잡도가 높으며 회귀간선을 가지고 있어 무한경로가 생성될 수 있는 복잡한 BPM들이다.

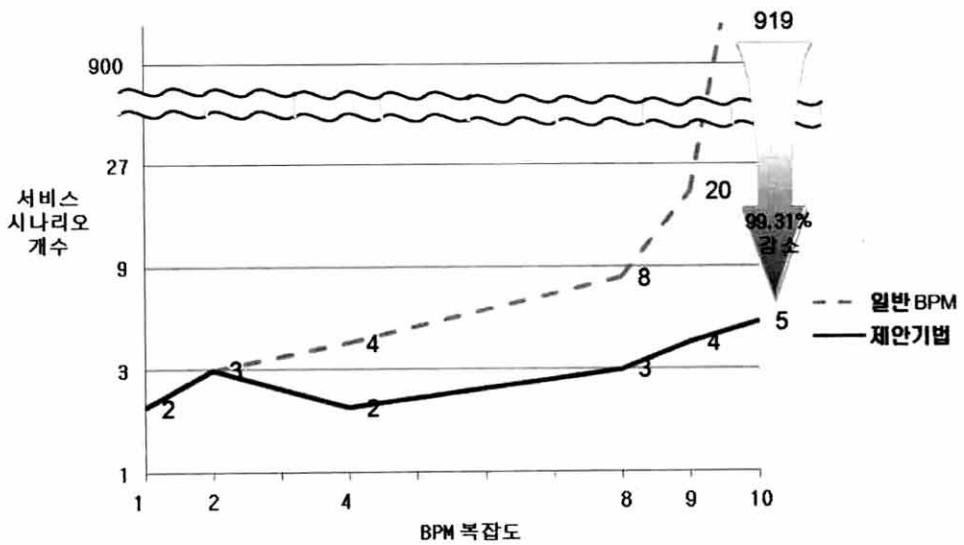
비교집합 2의 결과를 살펴보면 회귀간선과 그 회귀간선으로 인해 생성되는 순환경로에 포함되는 간선의 수가 제안 기법을 적용하지 않은 BPM에서 생성 가능한 서비스 시나리오 개수의 증가에 막대한 영향을 미치는 것을 알 수 있다. 특히 BPM 경로 복잡도 10의 사례는 순환경로 내부에 또 다른 순환경로가 있어 발생 가능한 서비스 시나리오의 경우의 수가 BPM 경로 복잡도 9에 비해서 기하급수적으로 증가하였고 경로를 1 회씩만 거칠 경우에도 생성 가능한 시나리오 개수가 919회 라는 큰 수치가 나온다. 그러나 이러한 증가율과 다르게 제안기법은 시나리오의 개수가 완만한 증가를 보이고, 시나리오 개수 감소 비율은 크게 증가하여 극히 복잡한 BPM 일수록 제안 기법으로 인한 효율성이 높아짐을 알 수 있다.

이러한 결과를 종합하였을 때 제안 기법은 기법을 적용하지 않은 경우와 비교하여 대단히 높은 효율성을 가지고 있음을 알 수 있다. 특히 BPM이 복잡해지고 순환경로가 많을수록 그 효율은 눈에 띄게 높아진다.

(그림 11)는 <표 4>의 결과를 그래프로 나타낸 것으로 BPM 경로 복잡도 10에서 급격하게 증가하는 서비스 시나리오 수를 표기하기 위하여 세로축인 서비스 시나리오의 개수를 3의 제곱수로 증가하도록 표기하였다. 가로축은 BPM 경로 복잡도를 나타낸다.

5.2 기존 연구와의 비교를 통한 정성적 평가

기존의 테스트 케이스 생성 기법 연구들은 제안 기법과 상이한 방식과 목표를 지니므로 직접적인 정량적 평가는 어렵기 때문에 비교를 통해 정성적 평가한다. 다음 <표 6>는 기존 연구와의 비교 평가 기준의 목록으로써 SOA의 특징을 포함하고 있는가를 평가하는 요소와 일반적인 테스트 케이스 생성 기법으로써의 평가 요소로 나뉘어서 평가한다.



(그림 11) 제안 기법 적용 결과 평가 그래프

〈표 6〉 기존 연구와의 비교 평가 기준

비교 항목		내용
SOA	비즈니스 관점 반영	비즈니스 관점이 테스트 케이스 생성에 영향을 미치는가?
	서비스 테스트 가능 유무	SOA의 논리 단위인 서비스나 서비스가 포함된 시스템을 테스트할 수 있는가?
	메시지 이동 정보 평가 가능 유무	서비스 시스템 내에서 메시지의 이동 정보를 체크하고 이를 테스트할 수 있는가?
사용성	사용의 편의성	해당 기법이 절차가 간단하고 사용하기 편리한가?
	자동화	해당 기법이 Tool이 지원되어 자동화 구현이 되는가?
	적용 도메인 범용성	특정 도메인에 한정적이지 않고 다양한 도메인에 적용이 가능한 범용성을 가지고 있는가?

SOA를 위한 테스트 케이스 생성 기법이라면 SOA 특징을 반영하고 있어야 하며, 따라서 3 가지의 평가 기준을 정하였다. 비즈니스 관점이 테스트 케이스 생성에 영향을 미치는지 여부는 기법의 입력이나 관련요소를 통해서 비즈니스 관점이 결과를 변화시킬 수 있는지를 판단하기 위한 기준이다. 비즈니스 관점이 기법의 입력이 될 경우엔 우수함, 관련 요소로써 작용하면 보통, 코드형태의 입력만 사용될 경우 미흡으로 판단한다. 서비스나 서비스 시스템을 테스트할 수 있는가를 평가하는 두 번째 기준은 상태가 존재하지 않아 FSM을 적용할 수 없는 문제를 극복할 수 있는가를 판단하기 위한 기준으로 상태에 상관 없으며 서비스를 테스트할 수 있다면 우수함, 상태가 영향은 미치지만 서비스를 테스트할 수 있다면 보통, 서비스에 적용은 가능하지만 그 특성을 무시하고 코드 수준의 테스트가 될 경우 미흡으로 판단한다. 마지막 메시지의 이동 경로를 확인하고 이를 평가할 수 있는가의 기준은 SOA의 느슨한 결합과 동적 재조합을 위한 평가 기준으로써 단위 테스트보다는 시스템 테스트적인 요소이며, 시스템 하위 단위간에 메시지의 이동 정보를 확인할 수 있다면 우수함, 시스템의 조합 상태를 확인할 수 있지만 하위 단위간의 메시지 이동 정보는 표현할 수 없다면 보통, 시스템 전체를 하나의 단위로 보고 테스트해야 한다면 미흡으로 평가한다.

일반적 테스트 케이스 생성 기법으로써 평가는 사용의 편의성과 효율을 위한 자동화, 그리고 범용적 사용가능성을 선택하였다. 사용의 편의성은 해당 기법이 절차가 간단하고 누구나 사용자들이 쉽게 이해가 가능하다면 우수함, 간편한 절차와 이해가 쉬운 두 가지 요소 중 한쪽만 만족한다면 보통, 기법이 복잡하고 전문가 수준의 노력이 필요할 경우엔 미흡으로 판단한다. 자동화는 테스트 케이스가 Tool이 구현되어 입력에 의해 자동으로 결과가 출력될 수 있다면 우수함, 알고리즘은 정의되었지만 Tool 이 없다면 보통, 자동화의 가능성은 보이지만 정의되지 않았을 경우에는 미흡이 된다. 범용성은 해당 기법이 특정 목적을 가진 도메인에 한정적이지 않고 널리 사용이 가능하다면 우수함, 기법의 입력이나 기법에 영향을 미치는 요소가 특정 도메인에만 사용되는 경우엔 보통, 입력은 물론 기법 자체의 방법도 특정 도메인에만 한정적일 경우에는 미흡으로 판단한다.

제안 기법과의 비교에 사용된 기존 연구들은 기존 SOA 테스트 케이스 기법 4개이다. 각 연구는 편의를 위해 각 기법의 대표 키워드를 이름으로 하여 Data Perturbation[11], WSDL Description[12], CV&V Framework[13], Decision Table[14]이라고 표기한다. <표 7>은 <표 6>의 평가 기준에 따라서 기존 연구와의 비교 평가한 결과를 보여준다.

각 연구에 관한 평가 값은 상기 언급한 평가 기준에 맞추어 제안 기법과의 상대적 평가를 통해 부여하였다. <표 7>의 평가 결과를 보면, 평가 항목 중 SOA의 특징과 관련된 비즈니스 관점을 반영한 부분에서는 기존의 SOA 테스트 기법들은 낮은 점수를 보인다. 기존 연구들은 XML 언어 기반의 소스 코드를 테스트 케이스 생성에 사용하여 비즈니스 관점을 반영하는 것에 대단히 미흡하다. 반면 모두 SOA를 위한 테스트 기법이므로 서비스를 테스트 하는 것은 우수함을 보인다. 그러나 서비스의 메시지 이동 정보는 다루고는

〈표 7〉 기존 연구와의 비교 평가 결과

비교 항목		Data Perturbation	WSDL Description	CV&V Framework	Decision Table	제안 기법
SOA	비즈니스 관점 반영	1	1	1	1	3
	서비스 테스트 가능 유무	3	3	3	3	3
	메시지 이동 정보 평가 가능 유무	3	2	2	2	3
사용성	사용의 편의성	2	2	2	3	2
	자동화	3	3	3	2	2
	적용 도메인 범용성	1	1	2	2	3
총 합		13	12	13	13	16

[범례] 3: 우수, 2: 보통, 1: 미흡

있으나 중점적인 사항이 아니라 보통으로 평가된다. 특히 Data Perturbation의 경우 우수함으로 판정되어 있으나 실제로 Peer-to-Peer의 두 종단간의 메시지만을 테스트하므로 복잡한 서비스 시스템에서는 제안 기법보다 부족한 모습을 보인다. 반면 제안 기법은 3가지에 있어 모두 우수하여 기존 SOA 테스트 기법들의 장점을 유지하면서도 단점을 개선하여 SOA 환경에 적용하기에 우수한 것으로 판단된다.

일반적인 테스트 케이스 기법으로써 기존 연구들은 자동화 구현이 되어 있어 높은 점수를 보이는 반면 적용 가능한 도메인에 한정적인 모습을 보이며 대부분 XML 언어를 사용하지 않는 시스템에는 적용이 힘들다. 반면 제안 기법은 높은 범용성을 가지고 있으나 기법의 적용 절차가 길어 편의성이 떨어지고 자동화 Tool 구현이 미흡해 아직 개선의 여지가 남았다고 볼 수 있다. 전체적으로 보았을 때에 총점이 16점으로 기존의 연구들에 비하여 우수한 편이라 볼 수 있다. 서비스를 테스트 하는 목적을 유지하면서 메시지 이동 정보를 평가할 수 있고 비즈니스 관점을 잘 반영하고 있다. 또한 서비스뿐만 아니라 BPMN으로 비즈니스 프로세스를 나타낼 수 있는 시스템이라면 본 기법을 적용할 수 있어 우수한 범용성을 보여준다. 이것은 본 논문의 목표를 잘 반영하고 있으며 기존의 연구 기법들보다 우수한 것으로 판단되어 기존의 문제점을 상당히 해결하였다고 볼 수 잇다.

6. 결론 및 향후 연구

테스트는 유지보수 및 그 비용과 관련하여 점점 중요성이 높아지고 있는 분야이며, 최근 주목을 받고 있는 SOA에서도 여전히 중요하게 여겨지고 있다. 그러나 전통적인 개발 방법론과 차이점을 가지고 있어 전통적 테스트 케이스 생성 기법들은 SOA의 특징을 반영하지 못하며, 기존의 SOA 테스트 기법들은 XML 언어에 한정적인 문제점을 가지고 있다.

본 논문에서는 SOA의 논리 단위인 서비스가 모여서 이루어진 서비스 시스템에 대한 테스트 케이스 생성 기법을 설계하고 이를 위하여 BPMN을 따라 작성된 BPM이 축약된 S-BPM을 사용한다. 이것은 비즈니스 관점으로 작성된 BPM이 테스트에 직접적인 영향을 미치게 되어 테스트 역시 비즈니스 관점으로 운영될 수 있는 장점을 가진다. 또한 BPM은 시스템에서 사용될 수 있는 모든 시나리오가 표기되므로 동적으로 재결합되어 다양한 서비스를 제공하는 서비스 시스템의 특성을 테스트하기에 적합하다. 그리고 S-BPM을 기반으로 방향성 그래프인 단순 서비스 경로 그래프를 생성하고 알고리즘을 적용하여 BPM에서 생성되는 테스트 케이스의 개수를 줄여 비즈니스 관점에서 운영되는 SOA에 있어서 더욱 부각되는 비용 감소를 만족한다. 이것은 최소개수로 구성되는 최단거리 테스트 케이스를 만들어 테스트를 실행하는 시나리오 수를 줄여 효율성을 높임으로써 가능하다. 마지막으로 기존의 SOA를 위한 테스트 케이스 생성 기법들이 XML 기반의 언어들을 사용하여 웹서비스에 한정적이었다면 비즈니스 프로세스를 사용하는 어떠한 시스템이라도 제안 기법을 적용할 수 있어 더욱 범용적인 사용이 가능하게 되었다. 이러한 기대 효과를 바탕으로 본 논문은 SOA의 특징을 잘 반영하면서도 효율성과 범용성이 높아 웹서비스 뿐만 아니라 SOC(Service-Oriented Computing)를 적용한 넓은 범위의 시스템에 사용하기에 적합하다 할 수 있다.

향후 연구로는 본 연구에서 제안한 기법을 프로그램으로 구현하여 자동화하고 변환 과정과 알고리즘을 개선하여 적용 비용을 더욱 줄이는 것과 BPMN과 직접적인 변환이 되는 BPEL(Business Process Execution Language)과의 연계를 통해서 테스트 케이스를 생성하는 기법의 연구를 진행하고자 한다.

참 고 문 헌

[1] Dongsu Kang, Chee-yang Song, Doo-kown Baik, "A Method of Service Identification For Product Line," IEEE Computer Society Press, International Conference on Convergence and Hybrid Information Technology (ICCIT '08), pp.1040-1045, Jul., 2008.

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

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

[4] 이승훈, 강동수, 송치양, 백두권, "메시지 흐름을 이용한 서비스의 테스트 케이스 생성 기법," 제31회 한국정보처리학회 춘계학술발표대회 논문집, 제16권 제1호, pp.420-423, 2009. 4.

[5] 이승훈, 강동수, 송치양, 백두권, "SOA를 위한 테스트 케이스 생성 기법," 제30회 한국정보처리학회 추계학술발표대회 논문집, 제15권 제2호, pp.527-530, 2008. 11.

[6] John D.McGregor, Timothy D.Korson, "Integrated object-oriented testing and development processes," Communications of the ACM, Vol.37, Issue9, pp.59-77, Sep., 1994.

[7] Tsuneo Yamaura, "How to design practical test cases," Hitachi Software Engineering, Dec., 1998.

[8] 전원영, 장수호, 김수동, "소프트웨어 시스템과 서비스 시스템의 유사성에 기반한 서비스 시스템 개발을 위한 체계적 설계 기법," 한국정보과학회 정보과학회논문지: 소프트웨어 및 응용, 제34권 제5호, pp.407-418, 2007. 5.

[9] 삼성SDS, "SOA방법론 - 서비스 식별 기법", 웹서비스/SOA 사업단, 2006.8.

[10] Matjaz B. Juric, Kapil Pant, "Business Process Driven SOA using BPMN and BPEL," Packt Publishing, Aug., 2008.

[11] Jeff Offutt, Wuzhi Xu, "Generating Test Cases for Web Services Using Data Perturbation", ACM SIGSOFT Software Engineering Notes archive, Vol.29, No.5, pp.1-10, Sep.,

2004.

[12] Xiaoying Bai, Wenli Dong, Wei-Tek Tsai, Yinong Chen, "WSDL-Based Automatic Test Case Generation for Web Services Testing," Proceedings of the IEEE International Workshop, pp.207-212, Oct., 2005.

[13] Hai Huang, Wei-Tek Tsai, Raymond Paul, Yinong Chen, "Automated Model Checking and Testing for Composite Web Services," Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, pp.300-307, 2005.

[14] Noikajana, S., Suwannasart, T., "Web Service Test Case Generation Based on Decision Table," 2008. QSIC '08. The Eighth International Conference, pp.321-326, Aug., 2008.

[15] Emig, C.; Weisser, J.; Abeck, Sebastian, "Development of SOA-Based Software Systems - an Evolutionary Programming Approach," In: Proc. of the Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services. IEEE, pp.182-188, Feb., 2006.

[16] 김건우, 이정화, 손진현, "비즈니스 프로세스 모델에서의 설계 이상 현상," 한국정보과학회 정보과학회논문지: 컴퓨팅의 실제 및 레터, 제14권 제9호, pp.850-863, 2008. 12.

[17] WfMC, "Workflow management coalition terminology & glossary(WFMC-TC-1011, Issue3.0)," Workflow Management Coalition, Feb., 1999.

[18] A. Aho, J. E. Hopcroft, J. D. Ullman, "The Design and Analysis of Computer Algorithms. Reading," MA: Addison-Wesley, 1974.

[19] Hao, R., Lee, D., Sinha, R. K., Griffeth, N., "Integrated System Interoperability Testing with Applications to VoIP," IEEE/ACM Transactions on Networking, Vol.12, Issue5, pp.23-836, 2004.

[20] 최은만, "컴퓨터를 이용한 시나리오 응용 방안," 1996년 한국정보처리학회 춘계학술발표 논문집, 제3권 제1호, pp.335-338, 1996.

[21] 김은주, 최은만, "시나리오를 이용한 객체 지향 시스템의 기능 테스트," 한국정보과학회 1996년도 가을 학술발표논문집, 제23권 제2호(B), pp.1523-1526, 1996. 10.

[22] 강성원, 이단형, 안유환, "수직적 추상의 도입에 의한 BPMN 추상기능의 확장," 정보처리학회논문지D, 제16-D권 제2호, pp.223-236, 2009. 4.

[23] Business Process Management Initiative, Business Process Modeling Notation version 1.0, http://206.222.18.10/media/documents/bpva10BPMNSpec/html/051.htm, May. 2004.

이 승 훈

e-mail : neatnesh@korea.ac.kr

2007년 단국대학교 전기전자컴퓨터학과 (학사)

2008년~현 재 고려대학교 컴퓨터·전파통신공학과 석사과정

관심분야 : 소프트웨어 공학, SOA, 테스팅

강 동 수

e-mail : greatkoko@hotmail.com

1997년 해군사관학교 전기공학(학사)

2006년 국방대학교 전산정보학과(석사)

2008년~현 재 고려대학교 컴퓨터·전파통신공학과 박사과정

관심분야 : 소프트웨어 공학, SOA, System Engineering, 요구공학

송 치 양

e-mail : cysong@knu.ac.kr

1985년 한남대학교 전산학과(학사)

1987년 중앙대학교 전산학과(이학석사)

2003년 고려대학교 컴퓨터학과(이학박사)

1990년~2005년 한국통신 중앙연구소 책임연구원

2005년~2007년 상주대학교 소프트웨어 공학과 조교수

2007년~현 재 경북대학교 소프트웨어공학과 조교수

관심분야 : UML 모델링 기술, 소프트웨어 개발방법, IP-TV 서비스

백 두 권

e-mail : baikdk@korea.ac.kr

1974년 고려대학교 수학과(학사)

1977년 고려대학교 산업공학과(공학석사)

1983년 Wayne State Univ. 전산학과(이학석사)

1985년 Wayne State Univ. 전산학과(이학박사)

2002년 고려대학교 정보통신대학 학장

1986년~현 재 고려대학교 컴퓨터학과 교수

1992년~현 재 ISO/IEC JTC1/SC32 국내위원회 위원장

1999년~현 재 정보통신진흥협회 데이터 기술위원회 의장

2005년~현 재 한국 시뮬레이션학회 고문

2009년~현 재 고려대학교 정보통신대학 학장

관심분야 : 데이터베이스, 소프트웨어공학, 데이터 공학, 컴포넌트 기반 시스템, 메타데이터 레지스트리, 정보 통합, 프로젝트 매니지먼트