KIPS Tr. Software and Data Eng.
Vol.6, No.3 pp.125~134 pISSN: 2287-5905

https://doi.org/10.3745/KTSDE.2017.6.3.125

A Method of Test Case Generation Using BPMN-Based Path Search

JeJun Park† · DongSu Kang††

ABSTRACT

The SOA (Service Oriented Architecture) based softwares are escalated because of quickly coping with business requirement. SOA can not apply to existing test method because of loosely coupled service and massage exchange architecture. In this paper, we suggest a method of test case generation using BPMN (Business Process Model and Notation). First we model processes, then make Business Flow Graph (BFG). After searching the euler path through symmetrized BFG about input and output degrees, we generate test cases. A method of test case generation using BPMN can apply at SOA-based system, and reduce the number of test cases.

Keywords : Software Testing, Generating Test Case, SOA, BPMN, BPEL

BPMN 기반 경로 탐색을 이용한 테스트 케이스 생성 기법

박 제 준† · 강 동 수††

요 약

SOA (Service Oriented Architecture)를 기반으로 하는 소프트웨어는 비즈니스 요구사항에 기민하게 대응할 수 있어 사용이 확대되고 있다. SOA는 느슨한 서비스 결합과 메시지 교환 중심 아키텍처라는 특징 때문에 기존의 테스팅 방법 적용이 제한된다. 본 논문에서는 BPMN (Business Process Model and Notation)을 이용해 테스트 케이스를 생성하는 기법을 제안한다. 먼저 프로세스를 모델링하고, 이를 통해 비즈니스 흐름 그래프(BFG)를 생성한다. BFG의 입출력 차수를 대칭하고, 오일러 경로를 탐색하여 테스트 케이스를 생성한다. BPMN을 이용한 테스트 케이스 생성 기법은 SOA 기반 시스템에 적용 가능하며, 테스트 케이스 수를 감소시킬 수 있다.

키워드 : 소프트웨어 테스팅, 테스트 케이스 생성, SOA, BPMN, BPEL

1. 서 론

SOA(Service Oriented Architecture)는 분산 어플리케이션의 서비스 구성을 기본 구조로 서비스 디자인, 서비스 식별, 서비스 관리, 서비스 구성과 서비스 테스팅 등 다양한 연구가 진행되고 있다[1, 2]. 소프트웨어 기업들이 비즈니스 요구사항을 연계하기 위해 SOA 기법을 사용한 솔루션을 사용하고 있으며, 이에 따른 기술의 발전으로 SOA 기반 시스템이 확대되고 있다.

비즈니스 환경의 변화에 기민하게 대응할 수 있는 SOA는 비즈니스 프로세스와 밀접한 관련을 가진다. SOA는 비즈니스를 연결된 서비스와 그 결과물로 통합하여 사용자의 요구사항을 충족하는 방법이기 때문이다. 따라서 각 비즈니스는 연결된 하나 혹은 그 이상의 서비스를 서로 연결하여 특정 기능을 수행할 수 있는 것을 말하게 되며, 이러한 서비스들이 애플리케이션과 연결되면서, 비즈니스와 애플리케이션이 상호 변화에 긴밀하게 대응할 수 있는 구조를 만들어 내는 것이다. SOA에서 비즈니스 프로세스는 BPMN (Business Process Model and Notation)이라는 표준 표기법으로 나타낼 수 있다. 이 표기법을 통해 추상적 비즈니스를 가시적인 워크플로우 형태로 인지할 수 있다.

이렇게 SOA를 기반으로 한 새로운 IT 환경에서 소프트웨어의 테스팅을 위해 필요한 테스트 케이스 생성은 기존의 CBD(Component-Based Development) 방법론이나 객체지향 방법론에서 사용하던 방법과 다르게 적용될 필요가 있다. 기존 개발방법론이 데이터 흐름 중심이므로 컴포넌트나 객체의 상태가 입력에 따라 변화되지만, SOA는 서비스의 느슨한 결합과 메시지 교환 중심 아키텍처, 상태를 정의하기 어려운 서비스라는 단위를 사용하기 때문에 기존 방법을 적용하기 어렵다. 서비스 기반 시스템의 대표적인 형태인 웹 서비스[3]에 대하여 테스트 케이스를 생성하는 기법[4, 5]이나 BPMN을 통해 테스트 케이스를 생성하는 기법[6-8] 등

† 준 회 원 : 해군238고속정편대장
†† 종신회원 : 국방대학교 컴퓨터공학전공 주임교수
Manuscript Received : September 20, 2016
Accepted : November 1, 2016
* Corresponding Author : DongSu Kang(greatkoko@kndu.ac.kr)

새로운 환경에서 테스팅을 위한 테스트 케이스 생성 기법에 대한 연구가 수행되었다.

본 논문에서는 SOA와 밀접한 관련을 가진 비즈니스 프로세스를 이용해 테스트 케이스를 생성하는 기법을 제안한다. BPMN을 사용하여 프로세스를 모델링하고, 이를 비즈니스 흐름 중심의 그래프로 변환한 후 각 노드의 입출력 차수 대칭을 통한 오일러 경로를 탐색한다. 제안된 기법을 사용함으로써 SOA를 기반으로 한 시스템에서 비즈니스 프로세스와 밀접한 테스팅이 가능할 것이다. 비즈니스 요구사항에 신속하게 대응하기 위한 SOA의 특징이 테스팅에 동일하게 나타나게 된다. 또한 BPMN 및 BPEL(Business Process Execution Language) 표준에 의해 비즈니스 프로세스를 이용한 테스트 케이스 생성 자동화 구현이 가능하며, 적은 수의 테스트 케이스로 전체 시스템을 테스팅 할 수 있다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 서비스 기반 체계에서 테스트 케이스를 생성하기 위한 연구에 대하여 소개한다. 3장에서 BPMN을 이용하여 테스트 케이스를 생성하기 위한 기법을 제시하고, 4장에서 적용 사례와 평가 및 검증을 수행한다. 5장에서 결론을 맺는다.

2. 관련 연구

서비스 기반 체계에 대한 테스트 케이스 생성 기법을 정리하면 Table 1과 같다. 서비스 기반 체계에서 테스트 케이스를 생성하는 기법이 공통적으로 나타내는 특징은 서비스에 대한 경로를 기반으로 테스트 케이스를 생성한다는 것이다. 이는 느슨하게 연결된 서비스의 집합이라는 SOA의 특징 때문에 각 서비스의 실행을 위한 경로를 탐색하는 방법이 적합하다고 판단한 결과이다.

Filippo Ricca & Paolo Tonella는 웹 애플리케이션을 UML 모델링으로 분석하고, 그래프 형태로 나타내어 모든 경로를 확인할 수 있는 테스트 케이스를 생성하는 방법을 연구하였다[4]. 그래프에서 각 노드의 순서 집합을 테스트 케이스로 설정하여 전체 시스템을 테스팅 할 수 있는 최소한의 테스트 케이스를 생성하는 방법이다. 기존 테스팅과 다르게 서비스 기반 웹 시스템에 적용 가능하다는 장점이 있지만 웹 애플리케이션에 국한되고, 테스트 케이스가 많아지는 단점이 있다.

ORD(Object Relationship Diagram)를 이용하여 웹을 표현하는 객체 관계를 확인하고 이를 바탕으로 OCFG(Object Control Flow Graph) 이용 모델화하여 테스트 케이스를 생성하는 방법[5]은 테스트 케이스의 감소라는 장점이 있다. 하지만 웹 서비스에 국한된다는 것과 모델링이 어렵다는 단점을 가지고 있다.

BPMN을 기반으로 모델 단축을 통해 테스트 케이스를 생성하는 기법[6]은 BPMN으로 모델링 된 프로세스에서 경로를 단축하여 테스트 케이스를 생성하는 방법이다. 웹 뿐만 아니라 서비스 기반 시스템에 전반적으로 적용 가능하므로 기존 웹 서비스에만 국한된 단점을 극복 가능하며, 단순화한 경로를 통해 테스트 케이스 감소가 가능하다. 하지만 순환 경로를 단순화하여 DAG(Directed Acyclic Graph)를 생성하는 과정에서 여러 개의 중복되는 순환에 대응하는데 제한되며, 동일한 서비스의 중복 수행이 발생 하는 문제가 있다. 또한 복잡한 변환 규칙을 자동화하여 수행하는데 제한을 가지고 있다.

BPMN으로부터 테스트 케이스를 생성하는 기법[7]은 BPMN에서 Flow Graph를 생성하여 DFS(Depth First Search)로 경로를 찾는 방법이다. 테스트 케이스 생성 방법을 단순화한 장점이 있지만, 경로 생성에 대한 구체화가 부족하고 루프 및 중복 수행에 대한 제한이 있다. 또한 DFS 경로 탐색으로 테스트 케이스가 많아진다.

비즈니스 프로세스를 활용한 테스트 케이스 생성 기법[8]은 BPMN에서 가능한 모든 경로를 찾아 내 결함 특성을 고려하여 테스트 케이스를 감소하는 방법이지만, 경로 생성 구체화 및 루프 대응이 제한된다.

모델링 언어는 다양하게 나타나지만, 표준화를 고려하고, 개발자 및 사용자 이해를 위한 모델링 언어를 사용할 필요가 있다. 또한 웹에 국한되는 문제 및 루프 및 중복 수행의 문제를 해결하면서, 테스트 케이스를 감소시키는 것이 효율적인 테스트 케이스 생성 기법이라 할 수 있다.

Table 1. Comparison of Test Case Generation Methods about Service Based System

Method	Modeling Language	Strong	Weakness
UML Modeling	UML	· Apply to web based system	· Limited to web based system · Increase in number of test case
Using ORD	OCFG	· Apply to web based system · Decrease in number of test case	· Limited to web based system · Difficult to model
BPMN with DAG	BPMN	· Apply to service based system · Decrease in number of test case	· Overlap with cycles & execution · Automatic implementation
BPMN with DFS	BPMN	· Apply to web based system.	· Embody in path generation · Increase in number of test case
Utilizing BPM	BPMN	· Apply to web based system. · Decrease in number of test case	· Embody in path generation · Overlap with cycles & execution

3. BPMN을 이용한 테스트 케이스 생성 기법

본 논문에서 BPMN을 기반으로 테스트 케이스를 생성하는 기법은 5가지 단계로 구성되며, Fig. 1과 같이 IDEF-0 표기법으로 나타낼 수 있다. 비즈니스 프로세스 모델링은 BPMN을 이용하여 비즈니스 프로세스를 모델링하는 것이며, 비즈니스 프로세스 모델(BPM)을 만들어 내는 단계이다. BFG(Business Flow Graph) 생성 단계에서는 BPMN을 그래프로 변환하는 규칙을 적용하여 BFG를 생성하고, 차수 대칭 단계에서는 차수 대칭(Degree Symmetry) 규칙을 적용하여 차수 대칭 그래프(D-S 그래프)를 생성한다. 이 그래프에서 오일러 경로 탐색을 이용한 테스트 경로를 생성한 후 테스트 케이스를 생성하게 된다. 첫 번째 단계인 비즈니스 프로세스 모델링은 표준 모델링 언어인 BPMN을 적용하게 모델링한다.

3.1 BFG(Business Flow Graph) 생성

BPMN이 풍부한 표기법을 가지고 있으므로 비즈니스 프로세스를 나타내는데 적합하지만, 이러한 표현 요소의 다양성은 그래프 형태로 변환을 어렵게 만드는 요소이기도 하다. 따라서 BPMN 요소 중 그래프에서 Node와 Edge가 될 수 있는 요소를 판단하여 그래프로 변환하는 것이 효과적이다. Table 2는 BPMN 요소가 그래프의 요소로 변환될 필요성을 분석한 결과이다. BPMN의 Event, Activity, Gateway는 실제로 수행되는 업무나 이벤트, 흐름을 분할하고 병합하는 역할을 하므로 그래프에서 Node로 사용될 필요가 있다. 그 외 요소는 흐름을 보조하고 표현을 구체화하는 요소로 그래프에서 불필요한 요소로 판단할 수 있다. 비즈니스 흐름을 그래프로 변환하는데 필요한 Node와 Edge로 구성된 그래프를 BFG(Business Flow Graph)로 정의한다.

Table 2. Necessity of Graph Element

BPMN Element		Graph Element
Flow	Event	Node
	Activity	Node
	Gateway	Node
Connecting	Sequence Flow	Edge
	Message Flow	Unnecessary
	Association	Unnecessary
	Data Association	Unnecessary
Data	Data Objects	Unnecessary
	Data Inputs	Unnecessary
	Data Outputs	Unnecessary
	Data Stores	Unnecessary
Swimlane	Pools	Unnecessary
	Lanes	Unnecessary
Artifact	Group	Unnecessary
	Text Annotation	Unnecessary

BPMN은 BPEL로 매핑되는 방법을 세부적으로 정의하고 있으며, 모델링 된 BPMN의 Sequence Flow를 BPEL로 변환한 상태를 바탕으로 비즈니스 Flow 그래프를 생성한다. 기본적으로 BPEL에서 Sequence Flow는 다음과 같은 표기법을 가진다.

```
< sequenceFlow   sourceRef="[source]"
                 targetRef="[target]"
                 name="[name]" id="[id]"/ >
```

모든 Sequence Flow는 Flow Object 사이 흐름을 나타내고 있으므로, 모든 source와 target은 Flow Object라고 할 수

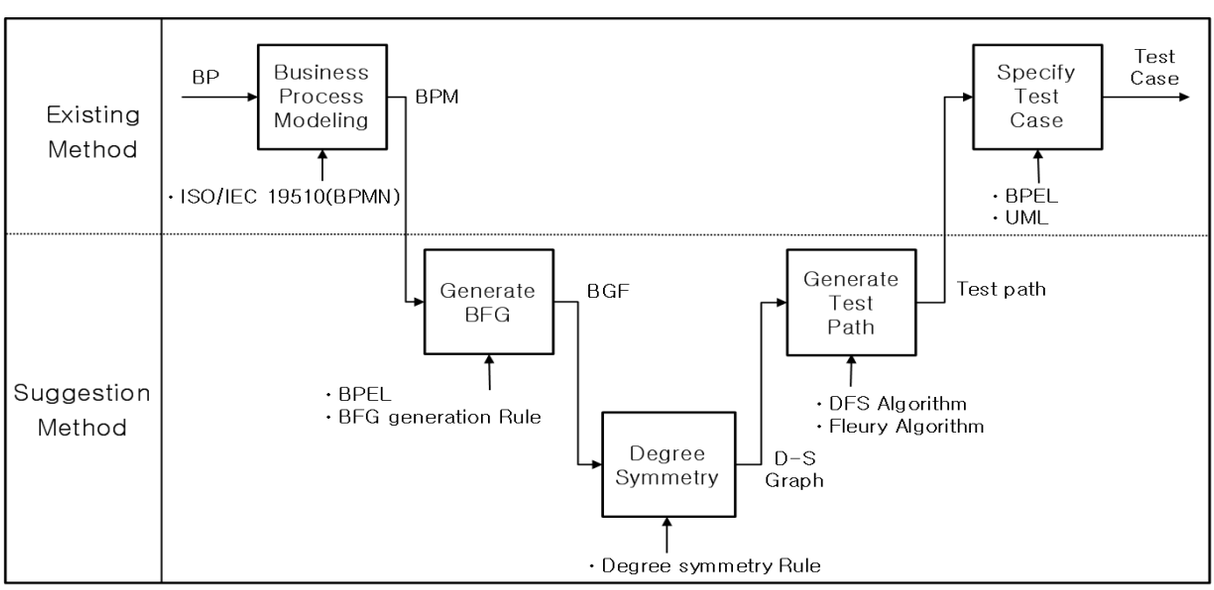

Fig. 1. A method of Test Case Generation Using BPMN

있다. 따라서 BPMN으로 모델링된 프로세스의 모든 Sequence Flow 집합으로 Flow Object가 Node, Sequence Flow가 Edge인 다이그래프를 생성할 수 있게 된다. 기존 연구에서도 BPMN을 기반으로 그래프를 통해 테스트 케이스를 생성하지만 그래프 변환 과정이 복잡하여 적용이 곤란하거나[6], 그래프 변환에 대한 구체화가 부족하여 복잡한 프로세스에서 적용이 제한된다[7]. BPEL에서 비즈니스 Flow 그래프를 생성하는 방법을 적용함으로써 단순화와 구현을 위한 구체화가 동시에 가능하다.

다만, Gateway는 흐름의 분기와 병합에 대하여 정의하고 있어 그래프에서 분기로 나타내는 것이 가능하나 Parallel Gateway의 경우 두 개 이상의 경로가 병렬로 수행되는 AND 형태이므로 Fig. 2와 같은 변환 규칙을 적용한다.

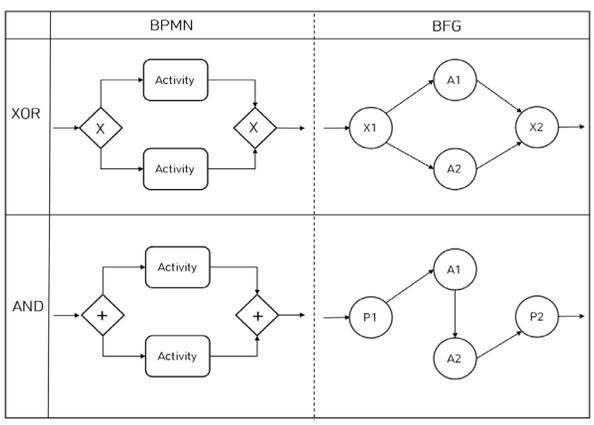

Fig. 2. Rules of Gateway Transformation

3.2 차수 대칭(Degree Symmetry)

BFG는 비즈니스 프로세스의 그래프 변환이므로, 모든 시작 노드는 Start Event, 모든 종료 노드는 End Event로 결정되어 있다. 따라서 하나의 그래프에 다수의 End Event가 있다면 End Event의 수만큼 그래프를 분할하여야 한다. 결정된 시작과 종료 노드 사이에 존재하는 중간 노드의 입/출력 차수가 모두 동일하다면 시작 노드와 종료 노드 사이에 오일러 경로가 존재한다.

시작 노드의 출력 차수와 종료 노드의 입력 차수는 항상 1이어야 한다. BFG에서 시작 노드의 출력 차수는 항상 1이므로, 차수 대칭으로 인해 시작 노드의 출력 차수 증가가 필요할 경우, 차수 대칭을 수행하지 않는다. 종료 노드의 입력 차수는 항상 1이어야 하므로, 차수 대칭으로 종료 노드의 입력 차수가 2 이상이 될 경우 해당 입력 차수만큼 종료 노드를 분할하여 종료 노드 입력 차수가 1인 BFG를 증가한다. Table 3의 차수 대칭 규칙을 적용하여 BFG로부터 D-S 그래프를 생성할 수 있다.

Table 3. Rules of Degree Symmetry

Node	Rules
Start Node (Start Event)	· Out-degree is always 1. (If out-degree change is necessary, don't apply a degree symmetry.)
End Node (End Event)	· In-degree is always 1. (If in-degree change is necessary, add to end node and BFG.)
Intermediate Node	· Symmetrize at bigger value between in-degree and out-degree. · Re-symmetrize after neighbor node degree changes.

3.3 테스트 경로 생성

모든 D-S 그래프는 출력 차수가 1인 하나의 시작 노드, 입력 차수가 1인 하나의 종료 노드를 가진 그래프이므로 오일러 경로가 존재한다. 다만, 차수 대칭을 실시하지 않은 D-S 그래프에는 오일러 경로가 존재하는지 알 수 없다. 따라서 Fig. 3과 같이 차수 대칭을 수행하지 않은 D-S 그래프는 DFS(Depth First Search) 알고리즘[9]으로 경로를 탐색하고, 차수대칭을 수행한 D-S 그래프는 Fleury 알고리즘[10]을 적용하여 찾아낸 오일러 경로가 테스트 경로가 된다.

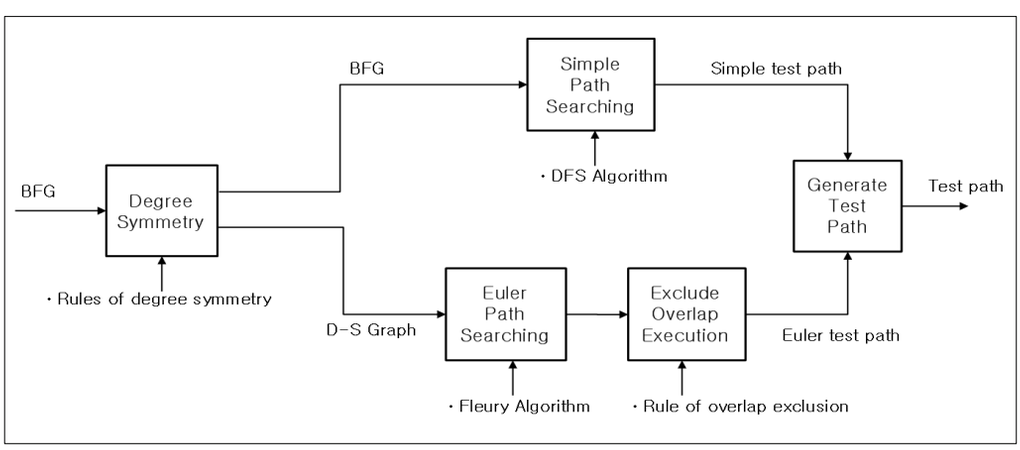

Fig. 3. Generation of Test Path

단순 경로 탐색을 통해 산출된 단순 테스트 경로는 고유한 테스트 경로이지만, 오일러 경로 탐색을 통해 산출된 테스트 경로는 동일한 중복 수행을 포함하고 있다. 따라서 이러한 중복 수행을 제거하기 위해 테스트 경로 생성 중 중복 수행을 제거하기 위한 조건에 따라 중복 수행을 제거한다.

중복 수행을 제거하기 위한 규칙은 단순하게 재방문하는 노드에 대한 생략이 아닌 중복된 경로에 대한 생략을 기본으로 한다. 따라서 중복 수행 제거 규칙은 이전 노드와 현재 노드, 다음 노드의 3가지 조건이 동일한 경우에만 생략하는 규칙을 적용한다. 예를 들어 A1→A2→A3→A4→A5→A2→A3→A4→A6 순 경로가 생성되었다면, 재방문 노드는 {A2, A3, A4} 이지만 A2는 이전 노드가 A5로 달라졌고, A4는 다음 노드가 A6로 달라졌으므로 생략하지 않고, 이전 노드와 다음 노드가 A2, A4로 동일한 A3 노드를 생략하여 A1→A2→A3→A4→A5→A2→A3(생략)→A4→A6 순 경로가 생성된다.

3.4 테스트 케이스 명세

테스트 경로 생성만으로 테스트 시나리오를 생성하여 테스트 케이스를 생성한 것으로 볼 수 있다. 기존의 연구[7, 8]는 테스트 경로 생성 후 테스트 케이스 생성은 테스트 케이스 명세를 위한 정의로 볼 수 있다. 또한 일부 연구에서는 테스트 경로를 이용하여 메시지 집합을 추출하여 테스트 케이스를 추출하는 방법[11]을 사용하기도 한다. 따라서 테스트 케이스 명세는 서비스 기반 체계의 테스팅 환경과 특성에 따라 기존 연구 또는 다양한 기법을 적용할 수 있으므로 생략한다.

4. 평가 및 검증

4.1 적용 사례

한국군은 NCOE(네트워크 중심 작전 환경, Network Centric Operational Environment)를 구축하고자 한다. 이를 위해 서비스 기반 시스템으로 변화가 요구되고, 비즈니스 프로세스의 중요성이 강조된다. 본 논문에서 테스트 케이스 생성을 위해 사용하는 사례는 한국군의 대공 표적 대응 프로세스로 해군과 공군의 합동작전을 위한 정보교환 및 교전 프로세스이다.

1) 비즈니스 프로세스 모델링

대공 표적 대응 프로세스는 Fig. 4와 같이 다양한 Activity와 Gateway, 많은 순환 경로를 보유하고 있다. 다양한 순환 경로에서 테스트 케이스를 효과적으로 생성할 수 있는지를 확인하기 위한 사례로 적절하다. BPMN모델링 툴은 일부 연구에서 이용된 Yaoqiang BPMN Editor[12]를 이용하였다.

2) BFG 생성

Name과 ID를 동일하게 사용할 수 있지만, 간소화를 위해 ID로 Table 4와 같이 변환하였다. 모델링 후 Sequence Flow를 Table 5와 같이 추출하였으며, 이를 통해 Fig. 5A의 BFG를 생성한다.

Table 4. ID - Name Mapping

ID	Name
S	Start Event
F1-2	End Event
A1	Contact target
A2	Identify target
A3	Report target
A4	Decision target ID
A5	Request target information
A6	Confirm target information
A7	Decision weapon
A8	Report weapon
A9	Change weapon
A10	Engage target
A11	Battle Damage Assessment
A12	Report result of engage
A13	Search target
X1-4	XOR Gateway
P1-4	AND Gateway

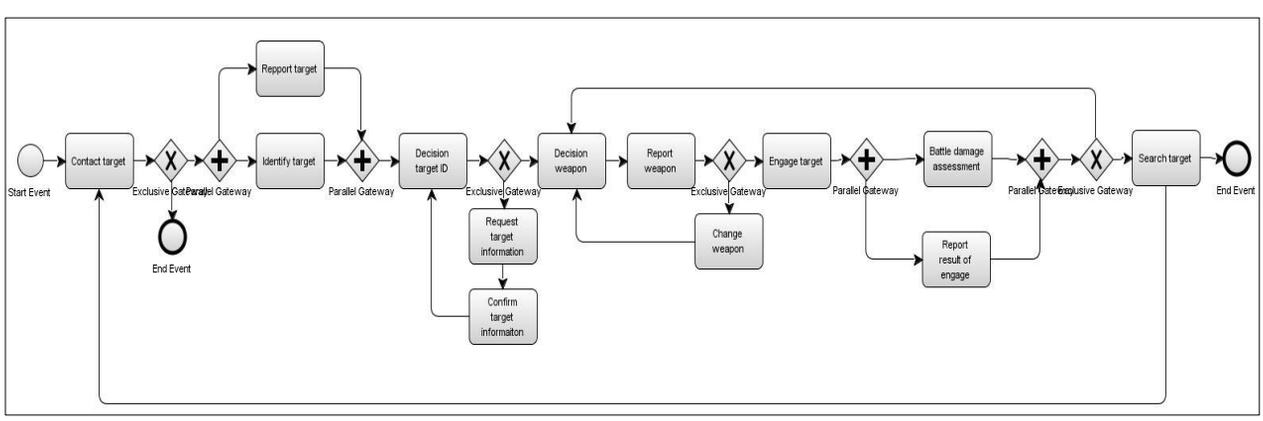

Fig. 4. Anti-air Target Engage Process

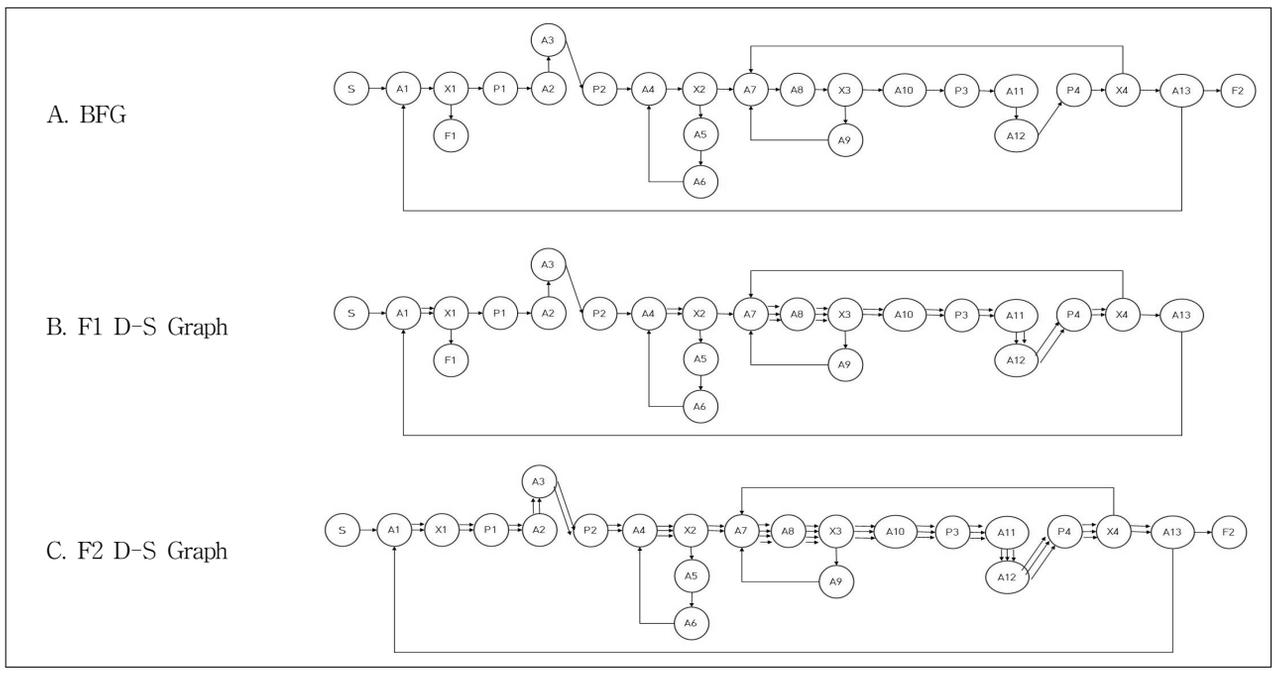

Fig. 5. BFG & D-S Graph

Table 5. Set of Sequence Flow

```
<sequenceFlow sourceRef="S" targetRef="A1" name="" id="1">
<sequenceFlow sourceRef="A1" targetRef="X1" name="" id="2">
<sequenceFlow sourceRef="X1" targetRef="F1" name="" id="3">
<sequenceFlow sourceRef="X1" targetRef="P1" name="" id="4">
<sequenceFlow sourceRef="P1" targetRef="A2" name="" id="5">
<sequenceFlow sourceRef="P1" targetRef="A3" name="" id="6">
<sequenceFlow sourceRef="A2" targetRef="P2" name="" id="7">
                    ⋮
```

3) 차수 대칭

BFG에서 종료 노드는 2개이므로 2개 이상의 D-S 그래프가 생성된다. 종료 노드로 진행하는 차수 대칭의 결과가 시작 노드의 차수를 변경하지 않으므로 Table 6의 결과에 따라 Fig. 5B, 5C와 같은 F1, F2 두 가지 종료 노드를 가지는 두 가지 D-S 그래프를 생성하게 된다.

4) 테스트 경로 생성

Table 7에서는 Fleury 알고리즘으로 찾아낸 오일러 경로를 나타내며, 중복 수행을 제거하기 위한 규칙을 적용하여 경로를 더욱 간소화 할 수 있다.

5) 테스트 케이스 명세

테스트 경로를 통해 테스트 시나리오를 생성하여 테스트 케이스를 명세할 수 있다. 이 분야는 다양한 연구가 있으므로 유사한 연구[7]에서 테스트 케이스 생성에 사용한 테스트 케이스 명세 기법을 적용한다. F1 경로에 대하여 테스트 케이스를 명세하면 Table 8과 같다. 흐름의 분기와 병합 역할을 수행하는 Gateway는 시나리오에서 생략한다.

Table 6. Result of Degree Symmetry

ID	F1 End Graph (Input, Output)		F2 End Graph (Input, Output)	
	BFG	D-S	BFG	D-S
S	(0, 1)			
A1	**(2, 1)**	**(2, 2)**	**(2, 1)**	**(2, 2)**
X1	**(1, 2)**	**(2, 2)**	**(1, 1)**	**(2, 2)**
P1	(1, 1)	(1, 1)	**(1, 1)**	**(2, 2)**
A2	(1, 1)	(1, 1)	**(1, 1)**	**(2, 2)**
A3	(1, 1)	(1, 1)	**(1, 1)**	**(2, 2)**
P2	(1, 1)	(1, 1)	**(1, 1)**	**(2, 2)**
A4	**(2, 1)**	**(2, 2)**	**(2, 1)**	**(3, 3)**
X2	**(1, 2)**	**(2, 2)**	**(1, 2)**	**(3, 3)**
A5	(1, 1)	(1, 1)	(1, 1)	(1, 1)
A6	(1, 1)	(1, 1)	(1, 1)	(1, 1)
A7	**(3, 1)**	**(3, 3)**	**(3, 1)**	**(4, 4)**
A8	**(1, 1)**	**(3, 3)**	**(1, 1)**	**(4, 4)**
X3	**(1, 2)**	**(3, 3)**	**(1, 2)**	**(4, 4)**
A9	(1, 1)	(1, 1)	(1, 1)	(1, 1)
A10	**(1, 1)**	**(2, 2)**	**(1, 1)**	**(3, 3)**
P3	**(1, 1)**	**(2, 2)**	**(1, 1)**	**(3, 3)**
A11	**(1, 1)**	**(2, 2)**	**(1, 1)**	**(3, 3)**
A12	**(1, 1)**	**(2, 2)**	**(1, 1)**	**(3, 3)**
P4	**(1, 1)**	**(2, 2)**	**(1, 1)**	**(3, 3)**
X4	**(1, 2)**	**(2, 2)**	**(1, 2)**	**(3, 3)**
A13	(1, 1)	(1, 1)	**(1, 2)**	**(2, 2)**
F1/F2	(1, 0)			

Table 7. Euler and Overlap Exclusion Path

Section		Path
F1	Euler path (39Node)	S, A1, X1, P1, A2, A3, P2, A4, X2, A5, A6, A4, X2, A7, A8, X3, A9, A7, A8, X3, A10, P3, A11, A12, P4, X4, A7, A8, X3, A10, P3, A11, A12, P4, X4, A13, A1, X1, F1
	Overlap exclusion path (31Node)	S, A1, X1, P1, A2, A3, P2, A4, X2, A5, A6, A4, X2, A7, A8, X3, A9, A7, **~~A8,~~** X3, A10, P3, A11, A12, P4, X4, A7, **~~A8~~**, **~~X3, A10, P3, A11, A12, P4,~~** X4, A13, A1, X1, F1
F2	Euler path (53Node)	S, A1, X1, P1, A2, A3, P2, A4, X2, A5, A6, A4, X2, A7, A8, X3, A9, A7, A10, P3, A11, A12, P4, X4, A7, A8, X3, A10, P3, A11, A12, P4, X4, A13, A1, X1, P1, A2, A3, P2, A4, X2, A7, A8, X3, A10, P3, A11, A12, P4, X4, A13, F2
	Overlap exclusion path (33Node)	S, A1, X1, P1, A2, A3, P2, A4, X2, A5, A6, A4, X2, A7, A8, X3, A9, A7, A10, P3, A11, A12, P4, X4, A7, A8, X3, A10, **P3**, **~~A11~~**, **~~A12~~**, **P4**, X4, A13, A1, **~~X1~~**, **P1**, **~~A2~~**, **~~A3~~**, **P2**, **~~A4~~**, **~~X2, A7, A8~~**, **~~X3~~**, **~~A10~~**, **~~P3, A11~~**, **~~A12~~**, **P4**, **~~X4~~**, A13, F2

Table 8. Test Cast Specification

Test Case ID	T01
Test Path	S→A1→A2→A3→A4→A5→A6→A4→A7→A8→A9→A7→A10→A11→A12→A7→A13→A1→F1
Input	Value
Output	
Expected test scenario	1. Start Event 2. Contact target 3. Identify target 4. Decision target ID 5. Request target information 6. Confirm target information 7. Decision target ID 8. Decision weapon 9. Report weapon 10. Change weapon 11. Decision weapon 12. Engage target 13. Battle Damage Assessment 14. Report result of engage 15. Decision weapon 16. Search target 17. Contact target 18. End Event

4.2 비교 평가

전체 시스템을 테스팅하기 위한 테스트 케이스는 적을수록 테스팅 효율이 높다. 따라서 동일한 프로세스를 기준으로 적용 기법 간 테스트 케이스 수를 비교함으로써 상대적인 테스팅 효율성을 평가할 수 있다. 또한 모든 Activity가 실행되는 것이 테스팅의 원칙이지만 동일한 Activity가 계속 중복 수행되는 것은 비효율 적이므로, 중복 수행 비율을 비교하면 상대적인 테스팅 효율을 비교할 수 있다. 비교 프로세스는 BPMN 예제[13]의 Stock Maintenance, The Nobel Prize Process, Shipment Process, The Travel Booking Diagram, E-Mail Voting Process의 5가지로 한다.

제안한 기법과 비교를 위해 BPMN 기반 모델 단축을 이용한 기법[6]과 BPMN으로 부터 생성된 그래프에서 DFS로 테스트 케이스 생성하는 기법[7] 두 가지 기법을 비교한다. 두 가지 기법은 BPMN을 기반으로 한다는 부분과 경로를 통해 시스템 테스트를 위한 테스트 케이스를 생성하므로 제시한 기법과 비교한다.

1) 테스트 케이스 수에 의한 평가

비교 프로세스는 흐름의 분기역할을 하는 Gateway와 순환 경로를 포함하고 있다. Gateway와 순환 경로가 많을수록 다양한 경로가 존재하는 복잡한 프로세스로 인지할 수 있다.

비교 프로세스는 Table 9에서와 같이 흐름 분기를 위한 요소, 순환 경로, 순환 복잡도 요소를 가지고 있다. 흐름을 분기하는 요소나 순환 경로가 다르더라도, 정량적으로 판단되는 복잡성이 동일한 경우도 있지만, 비교 프로세스는 순환 복잡도[14]가 모두 다른 프로세스이다.

Table 9. Process Comparison

Process	Split Gateway	Cycle	Cyclomatic Complexity
Stock Maintenance	1	0	1
Novel Prize	1	0	2
Shipment	3	0	4
Travel Booking	4	2	5
E-Mail Voting	6	3	6

BPMN으로부터 그래프 생성 후 DFS로 테스트 케이스를 생성하는 기법은 모든 Event, Gateway, Activity를 노드로 변환하여 그래프 생성한 후 DFS로 경로를 탐색한다. BPMN 기반 모델 단축 기법은 특정 Gateway와 Event 변환 규칙을 적용하여 그래프로 변환 후 SCC(Strongly Connected Component)를 별도로 분리하여 DAG(Directed Acyclic Graph)를 찾아내는 방법을 복합적으로 적용하여 경로를 생성한다. 제시한 기법과 다른 기법을 통해 생성한 비교 프로세스 테스트 케이스 수는 Table 10과 같다.

Table 10. Number of Test Case

Process	DFS	DAG	Suggested Method
Stock Maintenance	2	2	2
Novel Prize	2	2	2
Shipment	4	3	2
Travel Booking	15	9	6
E-Mail Voting	64	5	2

DFS를 이용한 경로 탐색은 AND/OR Gateway의 특성을 고려할 수 없으므로 모든 분기를 다른 경로로 탐색한다. 또한 순환 복잡도가 증가할수록 분기와 순환 경로의 수가 증가하고 있으며 이에 비례하여 테스트 케이스 수가 급격하게 증가하는 경향이 나타난다.

이러한 현상을 고려하여 BPMN 모델 단축 기법이 연구되었으나 다수의 순환 경로가 존재할 경우 순환 경로 사이에 대한 정의가 미흡하여 순환 경로의 중복으로 경로가 많아지는 경우가 발생한다.

본 논문에서 제안한 기법은 Gateway 특성을 고려하여 그래프로 변환하기 위한 규칙을 단순하게 적용할 수 있으며, Fig. 6에서와 같이 다른 기법에 비해 적은 수의 테스트 케이스를 생성한다. 논리적 복잡도인 순환 복잡도와 비례하여 테스트 케이스가 증가하지 않고 정해진 흐름을 단순한 방법으로 테스팅 할 수 있도록 한다. 따라서 복잡한 프로세스에서 상대적으로 더욱 효율적인 테스트 케이스를 생성할 수 있게 된다.

2) 중복 수행에 의한 평가

중복 수행은 동일한 노드를 얼마나 많이 재방문하는지 확인하는 방법을 적용한다. 전체 n개의 노드를 가진 그래프 G에서 모든 경로의 노드 수의 합과 n의 관계를 통해 중복 수행률 R(G) (1)의 공식으로 계산한다.

$$R(G) = (\sum_{k=1}^{k} P_k) / n \quad (1)$$

- n: 전체 노드 수
- k: 전체 경로 수
- P: 경로 노드 수

전체 노드가 한번만 수행되는 프로세스의 경우 중복 수행률은 1이고, 중복 수행 없이 가장 효율적으로 프로세스를 수행한 것으로 판단할 수 있으며, 중복 수행이 많을수록 R(G) 값이 증가한다. Table 11은 앞에서 테스트 케이스를 생성한 프로세스에 대하여 DFS, DAG, 제안 기법의 중복 수행률을 계산한 결과이다.

Table 11. Overlap Execution Rate

Process	Overlap execution rate		
	DFS	DAG	Suggested Method
Stock Maintenance	1.40	1.40	1.40
Novel Prize	1.77	1.70	1.77
Shipment	2.50	1.86	1.56
Travel Booking	13.00	7.20	3.67
E-Mail Voting	172.63	6.86	1.53

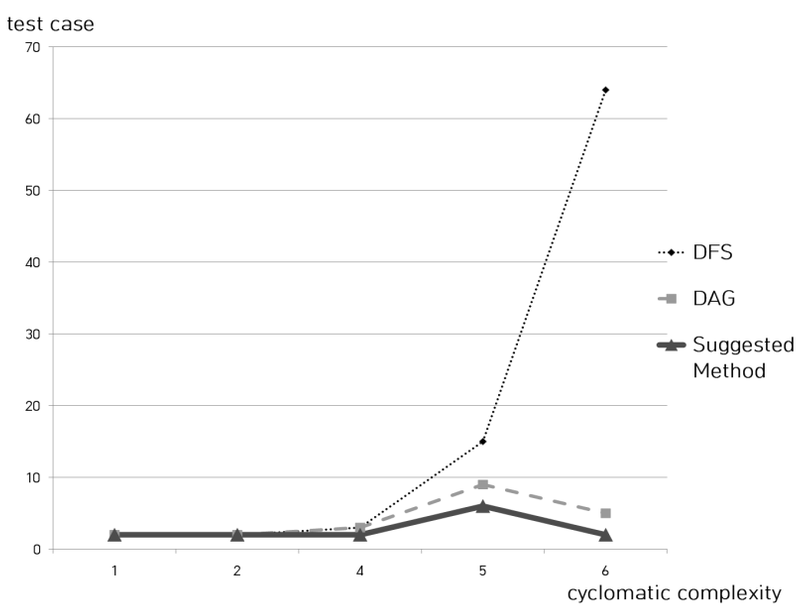

Fig. 6. Comparison of Test Case

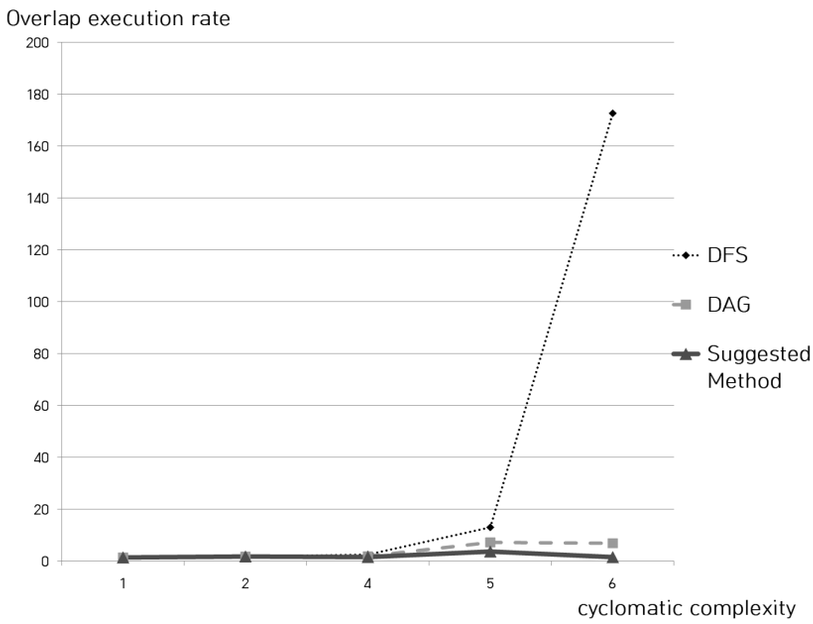

Fig. 7. Comparison of Overlap Execution Rate

테스트 케이스에서와 유사하게 DFS 방식의 경로 탐색은 다수의 경로에 의한 동일 노드 중복 수행률이 높으며 DAG 방식은 순환 경로가 많을수록 중복 수행률이 증가한다. 제안 기법은 Fig. 7과 같이 중복 수행률이 상대적으로 낮아 효율적이다.

5. 결론 및 향후 연구

SOA를 기반으로 하는 시스템의 확대에 따른 개발 기술이 지속적으로 발전하고 있으나, SOA를 위한 테스팅 기술은 상대적으로 연구가 미흡한 현실이다. SOA를 기반으로 하는 시스템의 경우 기존의 테스팅 기법을 적용하기 곤란한 특징을 가지고 있다. 또한, SOA의 확산은 비즈니스 변화에 대해 빨리 대응할 수 있으며, 테스팅 역시 비즈니스에 빠르게 대응할 필요가 있다.

본 논문에서는 BPMN을 이용한 테스트 케이스의 생성 기법을 제시하여 기존 테스팅 기법의 제한을 극복하고자 하였다. BPMN을 이용하여 그래프로 변환하고 차수 대칭을 통해 오일러 경로를 만들어 내므로 경로를 최소화하여 테스트 케이스를 감소시킬 수 있었다. 또한 중복 제거를 통한 동일 Activity 중복 수행을 최소화하여 효율적인 테스트 케이스 생성이 가능하다.

향후에는 서비스 지향의 특성을 고려하여, 경계값 검사나 커버리지 기반 등 여러 가지 테스트 케이스 생성 방법과 비교를 통해 적합성을 검증하고, 뮤테이션 테스트를 통해 테스트 케이스의 정확성을 확인할 예정이다.

References

[1] Dongsu Kang, Chee-yang Song, and Doo-Kwon Baik, “A Method of Service Identification for Product Line,” *IEEE Computer Society Press, International Conference on Convergence and Hybrid Information Technology(ICCIT08)*, pp.1040-1045, 2008.

[2] Dongsu Kang and Doo-Kwon Baik, “Bridge Software Product Lines and Service-Oriented Architectures for Service identification using BPM and FM,” *IEEE Computer Society Press, The 9th IEEE International Conference on Computer and Information Science(ICIS 2010)*, pp.755-759, 2010.

[3] Erl Thomas, “*Service-Oriented Architecture : Concepts, Technology, and Design*,” Prentice Hall, 2005.

[4] Filippo Ricca and Paolo Tonalla, “Analysis and Testing of Web Applications,” *Proceedings of the 23rd International Conference on Software Engineering, IEEE Computer Society*, 2001.

[5] Hyun Soo Kim and Eun Man Choi, “Effective Test Case Generation for Various Types of Web-based Software,” *The KIPS Transactions: Part* D, Vol.12, No.4, pp.569-582, 2005.

[6] Seung-Hoon Lee, Dongsu Kang, Cheeyang Song, and Dookwon Baik, “A Method of Test Case Generation using BPMN-based Model Reduction for Service System,” *The KIPS Transactions: Part* D, Vol.16, No.4, pp.595-612, 2009.

[7] Prat Yotyawilai and Taratip Suwannasart, “Design of a Tool for Generating Test Cases from BPMN,” in *Data and Software Engineering(ICODSE), IEEE 2014 International Conference on*, pp.1-6, 2014.

[8] Sarah Khader and Rana Yousef, “Utilizing Business Process Models to Generate Software Test Cases,” *IJCSI International Journal of Computer Science Issues*, Vol.13, Issue 2, March, 2016.

[9] Adam Drozdek, “Data Structures and Algorithms in C++,” 2nd ED., Course Technology, 2000.

[10] M. Fleury, “Deux problemes de geometrie de situation,” *Journal de Mathematiques Elementaires*, pp.257-261, 1883.

[11] Seung-Hoon Lee, Dongsu Kang, Cheeyang Song, and Dookwon Baik, “A Method of Test Case Generation using BPMN-based Model Reduction for Service System,” *KIPSD,* Vol.16, No.4, pp.595-612, 2009.

[12] Kajan, Ejub et al., “The network-based business process,” *IEEE Internet Computing*, 18.2, pp.63-69, 2014.

[13] OMG, “*BPMN 2.0 by Example Version 1.0*,” OMG Document Number: dtc/2010-06-02, June, 2010.

[14] T. J. McCabe, “A Complexity Measure,” *IEEE Trans. on Software Eng.*, Vol.2, No.4, 1976.

박 제 준

e-mail : bearjejun@gmail.com

2005년 해군사관학교(학사)

2016년 국방대학교 컴퓨터공학전공(석사)

2017년～현 재 해군238고속정편대장

관심분야 : Weapon System SW, SW Testing, SOA

강 동 수

e-mail : greatkoko@kndu.ac.kr

2011년 고려대학교 컴퓨터공학과(박사)

2012년 방위사업청 SW획득정책담당

2015년～현 재 국방대학교 컴퓨터공학전공 주임교수

관심분야 : Weapon System SW, SW Security Testing, SW Reliability, SW Process, SOA, Project management