



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2025-0051601
(43) 공개일자 2025년04월17일

(51) 국제특허분류(Int. Cl.) <i>G06F 11/36</i> (2025.01)	(71) 출원인 삼성에스디에스 주식회사 서울특별시 송파구 올림픽로35길 125 (신천동) 고려대학교 산학협력단 서울특별시 성북구 안암로 145, 고려대학교 (안암동5가)
(52) CPC특허분류 <i>G06F 11/3684</i> (2013.01) <i>G06F 11/3604</i> (2025.01)	(72) 발명자 문석현 서울특별시 송파구 올림픽로35길 125(신천동, 삼성SDS West Campus) 지윤찬 서울특별시 송파구 올림픽로35길 125(신천동, 삼성SDS West Campus) (뒷면에 계속)
(21) 출원번호 10-2024-0137854	(74) 대리인 특허법인충정
(22) 출원일자 2024년10월10일 심사청구일자 없음	
(30) 우선권주장 1020230134675 2023년10월10일 대한민국(KR)	

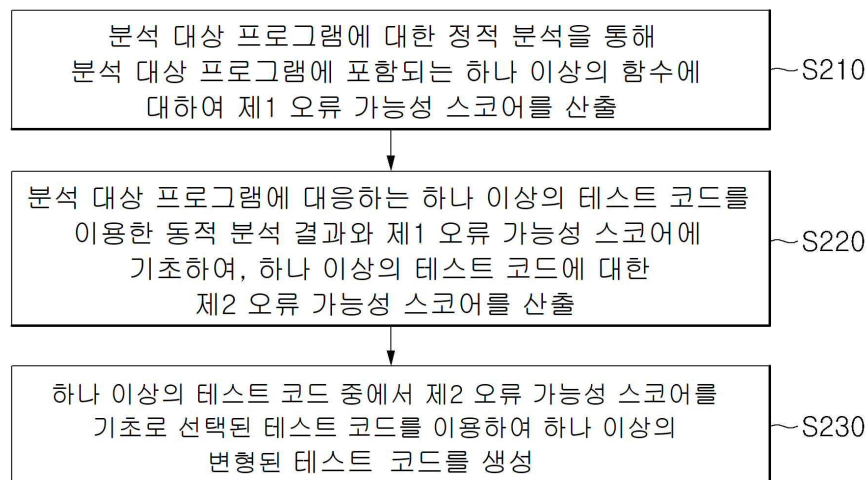
전체 청구항 수 : 총 20 항

(54) 발명의 명칭 **복합 분석을 활용한 테스트 코드 생성 방법, 장치, 시스템 및 컴퓨터 프로그램**

(57) 요약

본 발명은 정적 분석 및 동적 분석에 기반한 테스트 코드 생성 방법, 장치, 시스템 및 컴퓨터 프로그램에 관한 것으로서, 보다 구체적으로 본 발명에서는, 컴퓨팅 장치를 이용하여 테스트 코드를 생성하는 방법으로서, 분석 대상 프로그램에 대한 정적 분석을 통해 상기 분석 대상 프로그램에 포함되는 하나 이상의 함수에 대하여 제1 오류 가능성 스코어를 산출하는 단계; 상기 분석 대상 프로그램에 대응하는 하나 이상의 테스트 코드를 이용한 동적 분석 결과와 상기 제1 오류 가능성 스코어에 기초하여, 상기 하나 이상의 테스트 코드에 대한 제2 오류 가능성 스코어를 산출하는 단계; 및 상기 하나 이상의 테스트 코드 중에서 상기 제2 오류 가능성 스코어를 기초로 선택된 테스트 코드를 이용하여 하나 이상의 변형된 테스트 코드를 생성하는 단계를 포함하는, 방법을 개시한다.

대표도 - 도2



(52) CPC특허분류

G06F 11/3676 (2013.01)

G06F 11/3688 (2013.01)

G06F 11/3692 (2013.01)

G06F 11/3696 (2013.01)

(72) 발명자

오학주

서울특별시 강남구 압구정로 151, 122동 1002호 (압구정동, 현대아파트)

이명호

서울특별시 용산구 이촌로34길 29, 102동 1805호 (이촌동, 현대한강아파트)

박지성

서울특별시 서초구 효령로49길 57, 203동 503호(서초동, 서초2차이편한세상)

명세서

청구범위

청구항 1

컴퓨팅 장치를 이용하여 테스트 코드를 생성하는 방법으로서,

분석 대상 프로그램에 대한 정적 분석을 통해 상기 분석 대상 프로그램에 포함되는 하나 이상의 함수에 대하여 제1 오류 가능성 스코어를 산출하는 단계;

상기 분석 대상 프로그램에 대응하는 하나 이상의 테스트 코드를 이용한 동적 분석 결과와 상기 제1 오류 가능성 스코어에 기초하여, 상기 하나 이상의 테스트 코드에 대한 제2 오류 가능성 스코어를 산출하는 단계; 및

상기 하나 이상의 테스트 코드 중에서 상기 제2 오류 가능성 스코어를 기초로 선택된 테스트 코드를 이용하여 하나 이상의 변형된 테스트 코드를 생성하는 단계를 포함하는, 방법.

청구항 2

청구항 1에 있어서,

상기 제1 오류 가능성 스코어를 산출하는 단계는,

상기 하나 이상의 함수에 대하여 오류 발생이 가능한 표현식(expression) 위치에 도달할 수 있는 하나 이상의 후보 경로를 산출하는 단계;

상기 하나 이상의 후보 경로를 거쳐 상기 표현식에 도달하는 경우 오류 발생이 가능한 오류 가능 경로를 선별하는 단계; 및

상기 후보 경로 및 상기 오류 가능 경로를 기초로 상기 하나 이상의 함수에 대한 제1 오류 가능성 스코어를 도출하는 단계를 포함하는, 방법.

청구항 3

청구항 2에 있어서,

상기 하나 이상의 후보 경로를 산출하는 단계는,

상기 하나 이상의 함수에 대하여 오류 발생이 가능한 표현식을 식별하는 단계; 및

상기 분석 대상 프로그램에 대한 제어 흐름 그래프(CFG)를 기초로 상기 오류 발생이 가능한 표현식 위치에 도달할 수 있는 하나 이상의 후보 경로를 도출하는 단계를 포함하는, 방법.

청구항 4

청구항 2에 있어서,

상기 하나 이상의 후보 경로를 도출하는 단계에서는,

상기 오류 발생이 가능한 표현식 위치를 기준으로 상기 제어 흐름 그래프를 역방향으로 추적하여 상기 하나 이상의 후보 경로를 도출하는, 방법.

청구항 5

청구항 2에 있어서,

상기 오류 가능 경로를 선별하는 단계에서는,

상기 하나 이상의 후보 경로를 따라 각 문장(statement)을 실행하면서 오류 발생 여부를 판단하여 오류 가능 경로를 선별하는, 방법.

청구항 6

청구항 2에 있어서,

상기 제1 오류 가능성 스코어를 도출하는 단계는,

상기 후보 경로 중 상기 오류 가능 경로의 비율을 기초로 상기 하나 이상의 함수에 대한 경로 기준 오류 가능성 스코어를 산출하는 단계; 및

상기 하나 이상의 함수에 대한 경로 기준 오류 가능성 스코어에 상기 함수에서 호출되는 피호출 함수에 대한 경로 기준 오류 가능성 스코어를 합산하여 상기 함수에 대한 제1 오류 가능성 스코어를 산출하는 단계를 포함하는, 방법.

청구항 7

청구항 1에 있어서,

상기 제2 오류 가능성 스코어를 산출하는 단계에서는,

하나 이상의 테스트 코드를 이용한 동적 분석에서 호출된 하나 이상의 함수에 대한 각 제1 오류 가능성 스코어를 합산하여 상기 제2 오류 가능성 스코어를 산출하는, 방법.

청구항 8

청구항 7에 있어서,

상기 변형된 테스트 코드를 생성하는 단계에서는,

상기 하나 이상의 테스트 코드에 대한 각 제2 오류 가능성 스코어에 비례하여 높은 확률로 선택되어 변형된 테스트 코드로 생성되는, 방법.

청구항 9

청구항 1에 있어서,

상기 변형된 테스트 코드를 생성하는 단계에서는,

상기 하나 이상의 테스트 코드에 대한 정적 분석을 통해 오류 발생과 연관된 변수 또는 문장을 식별하고,

식별된 변수 또는 문장을 확률 기반으로 선택하여 변형된 테스트 코드를 생성하는, 방법.

청구항 10

청구항 1에 있어서,

상기 변형된 테스트 코드를 생성하는 단계에서는,

상기 변형된 테스트 코드를 기초로 동적 분석을 반복 수행하면서, 상기 제2 오류 가능성 스코어를 기초로 변형된 테스트 코드를 반복 생성하는, 방법.

청구항 11

프로세서; 및 메모리를 포함하는, 테스트 코드를 생성하는 장치에 있어서,

상기 메모리는 상기 프로세서에 의해 수행될 때 상기 장치가 특정 동작을 구현하도록 구성된 명령어를 포함하고, 상기 특정 동작은:

분석 대상 프로그램에 대한 정적 분석을 통해 상기 분석 대상 프로그램에 포함되는 하나 이상의 함수에 대하여 제1 오류 가능성 스코어를 산출하는 것;

상기 분석 대상 프로그램에 대응하는 하나 이상의 테스트 코드를 이용한 동적 분석 결과와 상기 제1 오류 가능성 스코어에 기초하여, 상기 하나 이상의 테스트 코드에 대한 제2 오류 가능성 스코어를 산출하는 것; 및

상기 하나 이상의 테스트 코드 중에서 상기 제2 오류 가능성 스코어를 기초로 선택된 테스트 코드를 이용하여 하나 이상의 변형된 테스트 코드를 생성하는 것을 포함하는, 장치.

청구항 12

청구항 11에 있어서,

상기 제1 오류 가능성 스코어를 산출하는 것에는,

상기 하나 이상의 함수에 대하여 오류 발생이 가능한 표현식(expression) 위치에 도달할 수 있는 하나 이상의 후보 경로를 산출하는 것;

상기 하나 이상의 후보 경로를 거쳐 상기 표현식에 도달하는 경우 오류 발생이 가능한 오류 가능 경로를 선별하는 것; 및

상기 후보 경로 및 상기 오류 가능 경로를 기초로 상기 하나 이상의 함수에 대한 제1 오류 가능성 스코어를 도출하는 것이 포함되는, 장치.

청구항 13

청구항 12에 있어서,

상기 하나 이상의 후보 경로를 산출하는 것에는,

상기 하나 이상의 함수에 대하여 오류 발생이 가능한 표현식을 식별하는 것; 및

상기 분석 대상 프로그램에 대한 제어 흐름 그래프(CFG)를 기초로 상기 오류 발생이 가능한 표현식 위치에 도달할 수 있는 하나 이상의 후보 경로를 도출하는 것이 포함되는, 장치.

청구항 14

청구항 12에 있어서,

상기 하나 이상의 후보 경로를 도출하는 것에서는,

상기 오류 발생이 가능한 표현식 위치를 기준으로 상기 제어 흐름 그래프를 역방향으로 추적하여 상기 하나 이상의 후보 경로를 도출하는, 장치.

청구항 15

청구항 12에 있어서,

상기 오류 가능 경로를 선별하는 것에서는,

상기 하나 이상의 후보 경로를 따라 각 문장(statement)을 실행하면서 오류 발생 여부를 판단하여 오류 가능 경로를 선별하는, 장치.

청구항 16

청구항 12에 있어서,

상기 제1 오류 가능성 스코어를 도출하는 것에는,

상기 후보 경로 중 상기 오류 가능 경로의 비율을 기초로 상기 하나 이상의 함수에 대한 경로 기준 오류 가능성 스코어를 산출하는 것; 및

상기 하나 이상의 함수에 대한 경로 기준 오류 가능성 스코어에 상기 함수에서 호출되는 피호출 함수에 대한 경로 기준 오류 가능성 스코어를 합산하여 상기 함수에 대한 제1 오류 가능성 스코어를 산출하는 것이 포함되는, 장치.

청구항 17

청구항 11에 있어서,

상기 제2 오류 가능성 스코어를 산출하는 것에서는,

하나 이상의 테스트 코드를 이용한 동적 분석에서 호출된 하나 이상의 함수에 대한 각 제1 오류 가능성 스코어

를 합산하여 상기 제2 오류 가능성 스코어를 산출하는, 장치.

청구항 18

청구항 11에 있어서,

상기 변형된 테스트 코드를 생성하는 것에서는,

상기 하나 이상의 테스트 코드에 대한 정적 분석을 통해 오류 발생과 연관된 변수 또는 문장을 식별하고,

식별된 변수 또는 문장을 확률 기반으로 선택하여 변형된 테스트 코드를 생성하는, 장치.

청구항 19

청구항 11에 있어서,

상기 변형된 테스트 코드를 생성하는 것에서는,

상기 변형된 테스트 코드를 기초로 동적 분석을 반복 수행하면서, 상기 제2 오류 가능성 스코어를 기초로 변형된 테스트 코드를 반복 생성하는, 장치.

청구항 20

프로세서에 의해 실행될 때 상기 프로세서를 포함하는, 테스트 코드를 생성하는 장치로 하여금 특정 동작을 구현하도록 구성된 명령어들을 저장하고 있는 컴퓨터 판독가능한 저장 매체에 있어서, 상기 특정 동작은:

분석 대상 프로그램에 대한 정적 분석을 통해 상기 분석 대상 프로그램에 포함되는 하나 이상의 함수에 대하여 제1 오류 가능성 스코어를 산출하는 것;

상기 분석 대상 프로그램에 대응하는 하나 이상의 테스트 코드를 이용한 동적 분석 결과와 상기 제1 오류 가능성 스코어에 기초하여, 상기 하나 이상의 테스트 코드에 대한 제2 오류 가능성 스코어를 산출하는 것; 및

상기 하나 이상의 테스트 코드 중에서 상기 제2 오류 가능성 스코어를 기초로 선택된 테스트 코드를 이용하여 하나 이상의 변형된 테스트 코드를 생성하는 것을 포함하는, 컴퓨터 판독가능한 저장 매체.

발명의 설명

기술 분야

[0001] 본 발명은 복합 분석을 활용한 테스트 코드 생성 방법, 장치, 시스템 및 컴퓨터 프로그램에 관한 것으로서, 보다 구체적으로는 분석 대상 프로그램에 대하여 정적 분석 및 동적 분석 기법에 기반하여 효율적으로 테스트 코드를 생성할 수 있는 복합 분석을 활용한 테스트 코드 생성 방법, 장치, 시스템 및 컴퓨터 프로그램에 관한 것이다.

배경 기술

[0002] 근래 여러 정보통신 서비스가 확산되면서 다양한 소프트웨어에 대한 테스트를 효율적으로 수행하기 위한 기법들이 시도되고 있다.

[0003] 이와 관련하여, 소프트웨어 개발 등의 과정에서 소프트웨어를 구성하는 각 단위 유닛 등에 대한 검증을 위한 테스트 코드를 생성하고 테스트를 수행하는 다양한 기법들이 활용되고 있다.

[0004] 그러나, 종래에는 통상적으로 코드 커버리지를 중심으로 문법 오류 또는 논리 오류 등을 테스트를 수행하게 되는데, 이러한 경우 문법 오류 등과 같은 단순 오류에 대해서는 효율적인 검증이 가능할 수 있으나, 널 포인트 예외(Null Point Exception, NPE) 등의 오류에 대한 취약점 탐지 능력은 코드 커버리지의 증가에 비례하여 개선되지 못하고 정체되거나, 코드 커버리지가 증가하더라도 오히려 탐지 능력이 저하되는 등의 문제가 따를 수 있었다.

[0005] 또한, 종래 기술에 따라 분석 대상 프로그램에 대한 소스 코드를 기반으로 정적 분석을 수행하거나, 생성된 테스트 코드를 기반으로 분석 대상 프로그램에 대한 동적 분석을 수행하더라도, 정적 분석 및 동적 분석에 의하여 개별적으로 취약점 탐지 결과를 도출할 수 있을 뿐, 정적 분석과 동적 분석을 동시에 활용하여 분석 대상 프로

그램의 취약점을 효율적으로 탐지하기는 어렵다는 제약이 따랐다.

[0006] 이에 따라, 소프트웨어 개발 등의 과정에서 보다 효율적으로 분석 대상 프로그램의 오류를 탐지할 수 있는 테스트 코드를 생성할 수 있도록 하는 방안이 요구되고 있으나, 아직 이에 대한 적절한 해법이 제시되지 못하고 있다.

선행기술문헌

특허문헌

[0007] (특허문헌 0001) 대한민국 공개특허 제10-2012-0063796호 (2012.06.18)

발명의 내용

해결하려는 과제

[0008] 본 발명은 상기와 같은 종래 기술의 문제점을 해결하기 위해 창안된 것으로, 분석 대상 프로그램에 대한 취약점을 탐지하기 위한 테스트 코드를 효율적으로 생성할 수 있는 테스트 코드 생성 방법, 장치, 시스템 및 컴퓨터 프로그램을 제공하는 것을 목적으로 한다.

[0009] 또한, 본 발명에서는 코드 커버리지가 증가하더라도 널 포인트 예외(NPE) 등의 오류에 대한 취약점 탐지 능력이 정제되거나 저하되는 문제를 개선할 수 있는 테스트 코드 생성 방법, 장치, 시스템 및 컴퓨터 프로그램을 제공하는 것을 목적으로 한다.

[0010] 나아가, 본 발명에서는 정적 분석과 동적 분석을 동시에 활용하여 분석 대상 프로그램의 취약점을 효율적으로 탐지할 수 있는 테스트 코드 생성 방법, 장치, 시스템 및 컴퓨터 프로그램을 제공하는 것을 목적으로 한다.

[0011] 본 발명에서 해결하고자 하는 기술적 과제는 이상에서 언급한 기술적 과제로 제한되지 않으며, 언급하지 않은 또 다른 기술적 과제는 본 명세서에 기재된 내용으로부터 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에게 명확하게 이해될 수 있을 것이다.

과제의 해결 수단

[0012] 본 발명의 제1 양상으로서, 컴퓨팅 장치를 이용하여 테스트 코드를 생성하는 방법으로서, 분석 대상 프로그램에 대한 정적 분석을 통해 상기 분석 대상 프로그램에 포함되는 하나 이상의 함수에 대하여 제1 오류 가능성 스코어를 산출하는 단계; 상기 분석 대상 프로그램에 대응하는 하나 이상의 테스트 코드를 이용한 동적 분석 결과와 상기 제1 오류 가능성 스코어에 기초하여, 상기 하나 이상의 테스트 코드에 대한 제2 오류 가능성 스코어를 산출하는 단계; 및 상기 하나 이상의 테스트 코드 중에서 상기 제2 오류 가능성 스코어를 기초로 선택된 테스트 코드를 이용하여 하나 이상의 변형된 테스트 코드를 생성하는 단계를 포함할 수 있다.

[0013] 여기서, 상기 제1 오류 가능성 스코어를 산출하는 단계는, 상기 하나 이상의 함수에 대하여 오류 발생이 가능한 표현식(expression) 위치에 도달할 수 있는 하나 이상의 후보 경로를 산출하는 단계; 상기 하나 이상의 후보 경로를 거쳐 상기 표현식에 도달하는 경우 오류 발생이 가능한 오류 가능 경로를 선별하는 단계; 및 상기 후보 경로 및 상기 오류 가능 경로를 기초로 상기 하나 이상의 함수에 대한 제1 오류 가능성 스코어를 도출하는 단계를 포함할 수 있다.

[0014] 또한, 상기 하나 이상의 후보 경로를 산출하는 단계는, 상기 하나 이상의 함수에 대하여 오류 발생이 가능한 표현식을 식별하는 단계; 및 상기 분석 대상 프로그램에 대한 제어 흐름 그래프(CFG)를 기초로 상기 오류 발생이 가능한 표현식 위치에 도달할 수 있는 하나 이상의 후보 경로를 도출하는 단계를 포함할 수 있다.

[0015] 또한, 상기 하나 이상의 후보 경로를 도출하는 단계에서는, 상기 오류 발생이 가능한 표현식 위치를 기준으로 상기 제어 흐름 그래프를 역방향으로 추적하여 상기 하나 이상의 후보 경로를 도출할 수 있다.

[0016] 또한, 상기 오류 가능 경로를 선별하는 단계에서는, 상기 하나 이상의 후보 경로를 따라 각 문장(statement)을 실행하면서 오류 발생 여부를 판단하여 오류 가능 경로를 선별할 수 있다.

[0017] 또한, 상기 제1 오류 가능성 스코어를 도출하는 단계는, 상기 후보 경로 중 상기 오류 가능 경로의 비율을 기초

로 상기 하나 이상의 함수에 대한 경로 기준 오류 가능성 스코어를 산출하는 단계; 및 상기 하나 이상의 함수에 대한 경로 기준 오류 가능성 스코어에 상기 함수에서 호출되는 피호출 함수에 대한 경로 기준 오류 가능성 스코어를 합산하여 상기 함수에 대한 제1 오류 가능성 스코어를 산출하는 단계를 포함할 수 있다.

- [0018] 이때, 상기 제2 오류 가능성 스코어를 산출하는 단계에서는, 하나 이상의 테스트 코드를 이용한 동적 분석에서 호출된 하나 이상의 함수에 대한 각 제1 오류 가능성 스코어를 합산하여 상기 제2 오류 가능성 스코어를 산출할 수 있다.
- [0019] 또한, 상기 변형된 테스트 코드를 생성하는 단계에서는, 상기 하나 이상의 테스트 코드에 대한 각 제2 오류 가능성 스코어에 비례하여 높은 확률로 선택되어 변형된 테스트 코드로 생성될 수 있다.
- [0020] 또한, 상기 변형된 테스트 코드를 생성하는 단계에서는, 상기 하나 이상의 테스트 코드에 대한 정적 분석을 통해 오류 발생과 연관된 변수 또는 문장을 식별하고, 식별된 변수 또는 문장을 확률 기반으로 선택하여 변형된 테스트 코드를 생성할 수 있다.
- [0021] 또한, 상기 변형된 테스트 코드를 생성하는 단계에서는, 상기 변형된 테스트 코드를 기초로 동적 분석을 반복 수행하면서, 상기 제2 오류 가능성 스코어를 기초로 변형된 테스트 코드를 반복 생성할 수 있다.
- [0022] 또한, 본 발명의 제2 양상으로, 프로세서; 및 메모리를 포함하는, 테스트 코드를 생성하는 장치에 있어서, 상기 메모리는 상기 프로세서에 의해 수행될 때 상기 장치가 특정 동작을 구현하도록 구성된 명령어를 포함하고, 상기 특정 동작은: 분석 대상 프로그램에 대한 정적 분석을 통해 상기 분석 대상 프로그램에 포함되는 하나 이상의 함수에 대하여 제1 오류 가능성 스코어를 산출하는 것; 상기 분석 대상 프로그램에 대응하는 하나 이상의 테스트 코드를 이용한 동적 분석 결과와 상기 제1 오류 가능성 스코어에 기초하여, 상기 하나 이상의 테스트 코드에 대한 제2 오류 가능성 스코어를 산출하는 것; 및 상기 하나 이상의 테스트 코드 중에서 상기 제2 오류 가능성 스코어를 기초로 선택된 테스트 코드를 이용하여 하나 이상의 변형된 테스트 코드를 생성하는 것을 포함할 수 있다.
- [0023] 여기서, 상기 제1 오류 가능성 스코어를 산출하는 것에는, 상기 하나 이상의 함수에 대하여 오류 발생이 가능한 표현식(expression) 위치에 도달할 수 있는 하나 이상의 후보 경로를 산출하는 것; 상기 하나 이상의 후보 경로를 거쳐 상기 표현식에 도달하는 경우 오류 발생이 가능한 오류 가능 경로를 선별하는 것; 및 상기 후보 경로 및 상기 오류 가능 경로를 기초로 상기 하나 이상의 함수에 대한 제1 오류 가능성 스코어를 도출하는 것이 포함될 수 있다.
- [0024] 또한, 상기 하나 이상의 후보 경로를 산출하는 것에는, 상기 하나 이상의 함수에 대하여 오류 발생이 가능한 표현식을 식별하는 것; 및 상기 분석 대상 프로그램에 대한 제어 흐름 그래프(CFG)를 기초로 상기 오류 발생이 가능한 표현식 위치에 도달할 수 있는 하나 이상의 후보 경로를 도출하는 것이 포함될 수 있다.
- [0025] 또한, 상기 하나 이상의 후보 경로를 도출하는 것에서는, 상기 오류 발생이 가능한 표현식 위치를 기준으로 상기 제어 흐름 그래프를 역방향으로 추적하여 상기 하나 이상의 후보 경로를 도출할 수 있다.
- [0026] 또한, 상기 오류 가능 경로를 선별하는 것에서는, 상기 하나 이상의 후보 경로를 따라 각 문장(statement)을 실행하면서 오류 발생 여부를 판단하여 오류 가능 경로를 선별할 수 있다.
- [0027] 또한, 상기 제1 오류 가능성 스코어를 도출하는 것에는, 상기 후보 경로 중 상기 오류 가능 경로의 비율을 기초로 상기 하나 이상의 함수에 대한 경로 기준 오류 가능성 스코어를 산출하는 것; 및 상기 하나 이상의 함수에 대한 경로 기준 오류 가능성 스코어에 상기 함수에서 호출되는 피호출 함수에 대한 경로 기준 오류 가능성 스코어를 합산하여 상기 함수에 대한 제1 오류 가능성 스코어를 산출하는 것이 포함될 수 있다.
- [0028] 이때, 상기 제2 오류 가능성 스코어를 산출하는 것에서는, 하나 이상의 테스트 코드를 이용한 동적 분석에서 호출된 하나 이상의 함수에 대한 각 제1 오류 가능성 스코어를 합산하여 상기 제2 오류 가능성 스코어를 산출할 수 있다.
- [0029] 또한, 상기 변형된 테스트 코드를 생성하는 것에서는, 상기 하나 이상의 테스트 코드에 대한 정적 분석을 통해 오류 발생과 연관된 변수 또는 문장을 식별하고, 식별된 변수 또는 문장을 확률 기반으로 선택하여 변형된 테스트 코드를 생성할 수 있다.
- [0030] 또한, 상기 변형된 테스트 코드를 생성하는 것에서는, 상기 변형된 테스트 코드를 기초로 동적 분석을 반복 수행하면서, 상기 제2 오류 가능성 스코어를 기초로 변형된 테스트 코드를 반복 생성할 수 있다.

[0031] 또한, 본 발명의 제3 양상으로, 프로세서에 의해 실행될 때 상기 프로세서를 포함하는, 테스트 코드를 생성하는 장치로 하여금 특정 동작을 구현하도록 구성된 명령어들을 저장하고 있는 컴퓨터 판독가능한 저장 매체에 있어서, 상기 특정 동작은: 분석 대상 프로그램에 대한 정적 분석을 통해 상기 분석 대상 프로그램에 포함되는 하나 이상의 함수에 대하여 제1 오류 가능성 스코어를 산출하는 것; 상기 분석 대상 프로그램에 대응하는 하나 이상의 테스트 코드를 이용한 동적 분석 결과와 상기 제1 오류 가능성 스코어에 기초하여, 상기 하나 이상의 테스트 코드에 대한 제2 오류 가능성 스코어를 산출하는 것; 및 상기 하나 이상의 테스트 코드 중에서 상기 제2 오류 가능성 스코어를 기초로 선택된 테스트 코드를 이용하여 하나 이상의 변형된 테스트 코드를 생성하는 것을 포함할 수 있다.

발명의 효과

[0032] 이에 따라, 본 발명의 일 실시예에 따른 복합 분석을 활용한 테스트 코드 생성 방법, 장치, 시스템 및 컴퓨터 프로그램에서는, 분석 대상 프로그램에 대한 취약점을 탐지하기 위한 테스트 코드를 효율적으로 생성할 수 있게 된다.

[0033] 또한, 본 발명의 일 실시예에 따른 복합 분석을 활용한 테스트 코드 생성 방법, 장치, 시스템 및 컴퓨터 프로그램에서는, 코드 커버리지가 증가하더라도 널 포인트 예외(NPE) 등의 오류에 대한 취약점 탐지 능력이 정제되거나 저하되는 문제를 개선할 수 있게 된다.

[0034] 나아가, 본 발명의 일 실시예에 따른 복합 분석을 활용한 테스트 코드 생성 방법, 장치, 시스템 및 컴퓨터 프로그램에서는, 정적 분석과 동적 분석을 동시에 활용하여 분석 대상 프로그램의 취약점을 효율적으로 탐지할 수 있게 된다.

[0035] 본 발명에서 얻을 수 있는 효과는 이상에서 언급한 효과로 제한되지 않으며, 언급하지 않은 또 다른 효과는 본 명세서에 기재된 내용으로부터 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자에게 명확하게 이해될 수 있을 것이다.

도면의 간단한 설명

[0036] 본 발명에 관한 이해를 돕기 위해 상세한 설명의 일부로 포함되는, 첨부도면은 본 발명에 대한 실시예를 제공하고, 상세한 설명과 함께 본 발명의 기술적 사상을 설명한다.

도 1은 본 발명의 일 실시예에 따른 테스트 코드 생성 시스템의 구성을 예시하는 도면이다.

도 2는 본 발명의 일 실시예에 따른 테스트 코드 생성 방법을 예시하는 순서도이다.

도 3은 본 발명의 일 실시예에 따른 테스트 코드 생성 장치의 구성과 동작을 설명하는 도면이다.

도 4는 본 발명의 일 실시예에 따른 제1 오류 가능성 스코어를 산출하는 단계에 대한 구체적인 순서도를 예시하는 도면이다.

도 5는 본 발명의 일 실시예에 따른 하나 이상의 후보 경로를 산출하는 단계에 대한 구체적인 순서도를 예시하는 도면이다.

도 6은 본 발명의 일 실시예에 따른 제1 오류 가능성 스코어를 도출하는 단계에 대한 구체적인 순서도를 예시하는 도면이다.

도 7 내지 도 8은 본 발명의 일 실시예에 따른 테스트 코드 생성 장치의 구체적인 동작을 예시하는 도면이다.

도 9는 본 발명의 일 실시예에 따른 테스트 코드 생성 방법의 구체적인 동작을 예시하는 순서도이다.

도 10은 본 발명의 일 실시예에 따른 테스트 코드를 생성하는 컴퓨팅 장치의 구성을 예시하는 도면이다.

발명을 실시하기 위한 구체적인 내용

[0037] 이하, 첨부된 도면을 참조하여 본 명세서에 개시된 실시예들을 상세히 설명한다. 본 발명의 목적, 특정한 장점들 및 신규한 특징들은 첨부된 도면들과 연관되어지는 이하의 상세한 설명과 바람직한 실시예들로부터 더욱 명확해질 것이다.

[0038] 이에 앞서 본 명세서 및 청구범위에 사용된 용어나 단어는 발명자가 그 자신의 발명을 가장 최선의 방법으로 설명하기 위해 개념을 적절하게 정의한 것으로 본 발명의 기술적 사상에 부합되는 의미와 개념으로 해석되어야 하

며, 단지 실시예들을 설명하기 위한 것일 뿐, 본 발명을 제한하는 것으로 해석되지 않아야 한다.

- [0039] 구성요소들에 참조 부호를 부여함에 있어, 참조 부호에 관계없이 동일하거나 유사한 구성요소는 동일한 참조 부호를 부여하고 이에 대한 중복되는 설명은 생략하기로 한다. 이하의 설명에서 사용되는 구성요소에 대한 접미사 "모듈" 및 "부"는 명세서 작성의 용이함을 고려하여 부여되거나 혼용되는 것으로서, 그 자체로 서로 구별되는 의미 또는 역할을 갖는 것은 아니며, 소프트웨어 또는 하드웨어 구성요소를 의미할 수 있다.
- [0040] 본 발명의 구성요소를 설명하는데 있어서, 단수 형태로 구성요소가 표현되는 경우 특별히 언급하지 않는 한 그 구성요소가 복수 형태도 포함하는 것으로 이해되어야 한다. 또한, "제1", "제2", 등의 용어는, 하나의 구성요소를 다른 구성요소와 구별하기 위해 사용되는 것으로, 구성요소가 상기 용어들에 의해 제한되는 것은 아니다. 또한, 어떤 구성요소가 다른 구성요소에 연결되는 경우, 구성요소와 다른 구성요소 사이에 또 다른 구성요소가 연결될 수도 있다는 것을 의미한다.
- [0041] 또한, 본 명세서에 개시된 실시예를 설명함에 있어서 관련된 공지 기술에 대한 구체적인 설명이 본 명세서에 개시된 실시예의 요지를 흐릴 수 있다고 판단되는 경우 그 상세한 설명을 생략한다. 또한, 첨부된 도면은 본 명세서에 개시된 실시예를 쉽게 이해할 수 있도록 하기 위한 것일 뿐, 첨부된 도면에 의해 본 명세서에 개시된 기술적 사상이 제한되지 않으며, 본 발명의 사상 및 기술 범위에 포함되는 모든 변경, 균등물 내지 대체물을 포함하는 것으로 이해되어야 한다.
- [0042] 이하에서는, 본 발명에 따른 복합 분석을 활용한 테스트 코드 생성 방법, 장치, 시스템 및 컴퓨터 프로그램의 예시적인 실시형태들을 첨부된 도면을 참조하여 상세히 설명한다.
- [0043] 먼저, 도 1에서는 본 발명의 일 실시예에 따른 테스트 코드 생성 시스템(100)의 구성 및 동작을 예시하고 있다. 도 1에서 볼 수 있는 바와 같이, 본 발명의 일 실시예에 따른 테스트 코드 생성 시스템(100)은, 하나 이상 사용자의 단말(110a, 110b) 및 상기 단말(110a, 110b)과 연동되어 사용자의 요청에 따라 테스트를 수행하고자 하는 대상 프로그램에 대한 테스트 코드를 생성하는 테스트 코드 생성 장치(120)를 포함하여 구성될 수 있다.
- [0044] 이때, 단말(110a, 110b)로서는 퍼스널 컴퓨터(PC), 노트북 PC, 태블릿 PC, 스마트폰, PDA 등 사용자가 테스트 코드 생성에 필요한 정보를 입력하거나 제공할 수 있는 다양한 단말이 사용될 수 있으나, 본 발명이 반드시 이에 한정되는 것은 아니며, 이외에도 대상 프로그램에 대한 정보를 보유하고 있는 서버 등 다양한 장치가 상기 단말(110a, 110b)로서 사용될 수 있다.
- [0045] 또한, 테스트 코드 생성 장치(120)는 하나 또는 둘 이상의 물리적 서버 장치(physical server)를 이용하여 구현될 수 있으나, 본 발명이 반드시 이에 한정되는 것은 아니며 이외에도 데스크톱 컴퓨터, 노트북, 태블릿, 스마트폰 등 개인용 전산 처리 장치를 이용하여 구성되거나, 클라우드 시스템을 기반으로 구성되거나, 나아가 전용 장치 등 다양한 형태로 구현하는 것도 가능하다.
- [0046] 나아가, 단말(110a, 110b)과 테스트 코드 생성 장치(120)를 하나의 서버로 결합하는 형태로 구현하는 것도 가능하다.
- [0047] 또한, 도 1에서 단말(110a, 110b)과 테스트 코드 생성 장치(120)를 연결하는 네트워크(130)로서는 유선 네트워크와 무선 네트워크 등을 사용할 수 있으며, 구체적으로 근거리 통신망 (LAN: Local Area Network), 도시권 통신망 (MAN: Metropolitan Area Network), 광역 통신망 (WAN: Wide Area Network) 등의 다양한 통신망을 포함할 수 있다. 또한, 상기 네트워크(130)는 공지의 월드 와이드 웹(WWW: World Wide Web)을 포함할 수도 있다. 나아가, 상기 네트워크(130)는 데이터 등을 송수신할 수 있도록 구성되는 데이터 버스(bus) 등을 이용하여 구현될 수도 있다.
- [0048] 또한, 도 2에서는 본 발명의 일 실시예에 따른 테스트 코드 생성 방법의 순서도를 예시하고 있다.
- [0049] 여기서, 도 2에 도시된 방법은 예를 들어, 테스트 코드 생성 장치(120)에 의해 수행될 수 있으며, 나아가 테스트 코드 생성 장치(120)는 도 10 및 도 10과 관련하여 후술하는 설명의 컴퓨팅 장치(50)를 포함하여 구현될 수 있다. 예컨대 테스트 코드 생성 장치(120)는 프로세서(10)를 구비하고 상기 프로세서(10)에서는 대상 프로그램에 대한 테스트 코드를 생성하기 위한 동작을 구현하도록 구성된 명령어를 수행할 수 있다.
- [0050] 보다 구체적으로, 도 2에서 볼 수 있는 바와 같이, 본 발명의 일 실시예에 따른 테스트 코드 생성 방법은, 컴퓨팅 장치(50)를 이용하여 테스트 코드를 생성하는 방법으로서, 분석 대상 프로그램에 대한 정적 분석을 통해 상기 분석 대상 프로그램에 포함되는 하나 이상의 함수에 대하여 제1 오류 가능성 스코어를 산출하는 단계(S210), 상기 분석 대상 프로그램에 대응하는 하나 이상의 테스트 코드를 이용한 동적 분석 결과와 상기 제1 오류 가능

성 스코어에 기초하여, 상기 하나 이상의 테스트 코드에 대한 제2 오류 가능성 스코어를 산출하는 단계(S220) 및 상기 하나 이상의 테스트 코드 중에서 상기 제2 오류 가능성 스코어를 기초로 선택된 테스트 코드를 이용하여 하나 이상의 변형된 테스트 코드를 생성하는 단계(S230)를 포함할 수 있다.

[0051] 여기서, 상기 제1 오류 가능성 스코어를 산출하는 단계(S210)는, 상기 하나 이상의 함수에 대하여 오류 발생이 가능한 표현식(expression) 위치에 도달할 수 있는 하나 이상의 후보 경로를 산출하는 단계(S211), 상기 하나 이상의 후보 경로를 거쳐 상기 표현식에 도달하는 경우 오류 발생이 가능한 오류 가능 경로를 선별하는 단계(S212) 및 상기 후보 경로 및 상기 오류 가능 경로를 기초로 상기 하나 이상의 함수에 대한 제1 오류 가능성 스코어를 도출하는 단계(S213)를 포함할 수 있다.

[0052] 또한, 상기 하나 이상의 후보 경로를 산출하는 단계(S211)는, 상기 하나 이상의 함수에 대하여 오류 발생이 가능한 표현식을 식별하는 단계(S2111) 및 상기 분석 대상 프로그램에 대한 제어 흐름 그래프(CFG)를 기초로 상기 오류 발생이 가능한 표현식 위치에 도달할 수 있는 하나 이상의 후보 경로를 도출하는 단계(S2112)를 포함할 수 있다.

[0053] 또한, 상기 하나 이상의 후보 경로를 도출하는 단계(S211)에서는, 상기 오류 발생이 가능한 표현식 위치를 기준으로 상기 제어 흐름 그래프를 역방향으로 추적하여 상기 하나 이상의 후보 경로를 도출할 수 있다.

[0054] 또한, 상기 오류 가능 경로를 선별하는 단계(S212)에서는, 상기 하나 이상의 후보 경로를 따라 각 문장(statement)을 실행하면서 오류 발생 여부를 판단하여 오류 가능 경로를 선별할 수 있다.

[0055] 또한, 상기 제1 오류 가능성 스코어를 도출하는 단계(S213)는, 상기 후보 경로 중 상기 오류 가능 경로의 비율을 기초로 상기 하나 이상의 함수에 대한 경로 기준 오류 가능성 스코어를 산출하는 단계(S2131) 및 상기 하나 이상의 함수에 대한 경로 기준 오류 가능성 스코어에 상기 함수에서 호출되는 피호출 함수에 대한 경로 기준 오류 가능성 스코어를 합산하여 상기 함수에 대한 제1 오류 가능성 스코어를 산출하는 단계(S2132)를 포함할 수 있다.

[0056] 이때, 상기 제2 오류 가능성 스코어를 산출하는 단계(S220)에서는, 하나 이상의 테스트 코드를 이용한 동적 분석에서 호출된 하나 이상의 함수에 대한 각 제1 오류 가능성 스코어를 합산하여 상기 제2 오류 가능성 스코어를 산출할 수 있다.

[0057] 또한, 상기 변형된 테스트 코드를 생성하는 단계(S230)에서는, 상기 하나 이상의 테스트 코드에 대한 각 제2 오류 가능성 스코어에 비례하여 높은 확률로 선택되어 변형된 테스트 코드로 생성될 수 있다.

[0058] 또한, 상기 변형된 테스트 코드를 생성하는 단계(S230)에서는, 상기 하나 이상의 테스트 코드에 대한 정적 분석을 통해 오류 발생과 연관된 변수 또는 문장을 식별하고, 식별된 변수 또는 문장을 확률 기반으로 선택하여 변형된 테스트 코드를 생성할 수 있다.

[0059] 또한, 상기 변형된 테스트 코드를 생성하는 단계(S230)에서는, 상기 변형된 테스트 코드를 기초로 동적 분석을 반복 수행하면서, 상기 제2 오류 가능성 스코어를 기초로 변형된 테스트 코드를 반복 생성할 수 있다.

[0060] 이에 따라, 본 발명의 일 실시예에 따른 복합 분석을 활용한 테스트 코드 생성 방법, 장치, 시스템 및 컴퓨터 프로그램에서는, 분석 대상 프로그램에 대한 취약점을 탐지하기 위한 테스트 코드를 효율적으로 생성할 수 있고, 또한 코드 커버리지가 증가하더라도 널 포인트 예외(NPE) 등의 오류에 대한 취약점 탐지 능력이 정제되거나 저하되는 문제를 개선할 수 있으며, 나아가 정적 분석과 동적 분석을 동시에 활용하여 분석 대상 프로그램의 취약점을 효율적으로 탐지할 수 있게 된다.

[0061] 이하, 각 도면을 참조하여 본 발명의 일 실시예에 따른 테스트 코드 생성 시스템(100)의 구성 및 동작을 보다 자세하게 살핀다.

[0062] 먼저, S210 단계에서는, 테스트 코드 생성 장치(120) 등 컴퓨팅 장치(50)에서 분석 대상 프로그램에 대한 정적 분석을 통해 상기 분석 대상 프로그램에 포함되는 하나 이상의 함수에 대하여 제1 오류 가능성 스코어를 산출하게 된다.

[0063] 여기서, 분석 대상 프로그램에는 소프트웨어를 구성하는 단위 유닛 등이 포함될 수 있으나, 본 발명이 반드시 이에 한정되는 것은 아니다.

[0064] 또한, 정적 분석(static analysis)이라 함은 분석하고자 하는 대상 프로그램을 실행하지 않고 소스 코드 또는 바이너리 상태에서 분석하는 기법을 말하며, 보다 구체적으로 요약 해석(abstract interpretation) 기법, 데이

터 종속(data dependency) 기법 등을 포함할 수 있으나, 본 발명이 반드시 이에 한정되는 것은 아니다.

- [0065] 또한, 본 발명에서 함수에 대한 제1 오류 가능성 스코어라 함은 상기 함수에서 널 포인터 예외(Null Pointer Exception, NPE) 오류 등 미리 정의된 오류가 발생할 확률을 의미할 수 있으며, 아래에서는 널 포인터 예외(NPE) 오류를 중심으로 본 발명의 구체적인 구성 및 동작을 설명하나, 이는 본 발명의 일 실시예에 해당할 뿐 본 발명이 반드시 이에 한정되는 것은 아니다.
- [0066] 또한, 본 발명에서 함수에는 프로그래밍 언어 등에서 사용되는 통상적인 함수(function)에 더하여 메서드(method), 프로시저(procedure) 등이 포함될 수 있으며, 나아가 이외에도 호출되거나 참조되는 등의 형식으로 수행되면서 널 포인터 예외(NPE) 등의 오류를 유발할 수 있는 다양한 형식의 프로그래밍 블록이 모두 포함될 수 있다.
- [0067] 보다 구체적인 예를 들어, 도 3을 참조하여 살펴보면, 본 발명의 일 실시예에 따른 테스트 코드 생성 장치(120)는 시드 생성 블록(210) 및 변형 기반 테스트 블록(220)을 포함하여 구성될 수 있다.
- [0068] 이때, 도 3에서 볼 수 있는 바와 같이, 상기 시드 생성 블록(210)에서는 JAVA 등 다양한 프로그래밍 언어로 작성된 분석 대상 프로그램에 대한 정적 분석(static analysis)(211)을 통해 상기 분석 대상 프로그램에 포함되는 하나 이상의 함수(보다 구체적으로, 함수나 메서드 혹은 프로시저 등)에 대하여 널 포인터 예외(NPE) 발생 가능성(예를 들어, 도3에서 NPE-likely Score (per method)) 등의 제1 오류 가능성 스코어를 산출할 수 있다.
- [0069] 또한, S220 단계에서는, 상기 분석 대상 프로그램에 대응하는 하나 이상의 테스트 코드를 이용한 동적 분석 결과와 상기 제1 오류 가능성 스코어에 기초하여, 상기 하나 이상의 테스트 코드에 대한 제2 오류 가능성 스코어를 산출하게 된다.
- [0070] 보다 구체적인 예를 들어, 도 3에서 볼 수 있는 바와 같이, 상기 시드 생성 블록(210)의 테스트 코드 생성 모듈(212)을 통해 상기 분석 대상 프로그램에 대응하는 하나 이상의 테스트 코드를 생성하고, 또한 상기 하나 이상의 테스트 코드에 대한 동적 분석(dynamic analysis)를 통해 상기 하나 이상의 테스트 코드와 그에 대한 제2 오류 가능성 스코어(예를 들어, 도3에서 Seed Test Case with NPE-likely Score 등)를 산출할 수 있다.
- [0071] 다음으로, S230 단계에서는, 상기 하나 이상의 테스트 코드 중에서 상기 제2 오류 가능성 스코어를 기초로 선택된 테스트 코드를 이용하여 하나 이상의 변형된 테스트 코드를 생성할 수 있다.
- [0072] 보다 구체적인 예를 들어, 도 3에서 볼 수 있는 바와 같이, 상기 변형 기반 테스트 블록(220)의 시드 선택 모듈(221)을 이용하여 상기 제2 오류 가능성 스코어를 기초로 하나 이상의 테스트 코드를 선택할 수 있으며, 이어서 테스트 코드 변형 모듈(222)을 이용하여 상기 하나 이상의 테스트 코드를 기초로 하나 이상의 변형된 테스트 코드(예를 들어, 도 3의 Mutant)를 생성할 수 있다.
- [0073] 나아가, 상기 변형 기반 테스트 블록(220)에서는 상기 변형된 테스트 케이스를 기초로 동적 분석(dynamic analysis)을 반복 수행하면서, 상기 제2 오류 가능성 스코어를 기초로 변형된 테스트 케이스를 반복 생성하는 것이 가능하다.
- [0074] 또한, 도 4에서 볼 수 있는 바와 같이, 상기 제1 오류 가능성 스코어를 산출하는 단계(S210)는, 상기 하나 이상의 함수에 대하여 오류 발생이 가능한 표현식(expression) 위치에 도달할 수 있는 하나 이상의 후보 경로를 산출하는 단계(S211), 상기 하나 이상의 후보 경로를 거쳐 상기 표현식에 도달하는 경우 오류 발생이 가능한 오류 가능 경로를 선별하는 단계(S212) 및 상기 후보 경로 및 상기 오류 가능 경로를 기초로 상기 하나 이상의 함수에 대한 제1 오류 가능성 스코어를 도출하는 단계(S213)를 포함할 수 있다.
- [0075] 보다 구체적으로, 도 5에서 볼 수 있는 바와 같이, 상기 시드 생성 블록(도 3의 210)에서는 주어진 분석 대상 프로그램(310)의 하나 이상의 함수에 대하여 오류 발생이 가능한 표현식(expression) 위치에 도달할 수 있는 하나 이상의 후보 경로를 산출하고(320), 상기 하나 이상의 후보 경로를 거쳐 상기 표현식에 도달하는 경우 오류 발생이 가능한 오류 가능 경로를 선별한 후(330), 상기 후보 경로 및 상기 오류 가능 경로를 기초로 상기 하나 이상의 함수에 대한 제1 오류 가능성 스코어를 도출하는 것이 가능하다(340).
- [0076] 여기서, 상기 하나 이상의 후보 경로를 도출하는 단계(S211)에서는, 상기 오류 발생이 가능한 표현식 위치를 기준으로 상기 제어 흐름 그래프를 역방향으로 추적하여 상기 하나 이상의 후보 경로를 도출할 수 있다.
- [0077] 또한, 상기 오류 가능 경로를 선별하는 단계(S212)에서는, 상기 하나 이상의 후보 경로를 따라 각 문장(statement)을 실행하면서 오류 발생 여부를 판단하여 오류 가능 경로를 선별할 수 있다.

- [0078] 나아가, 도 6에서 볼 수 있는 바와 같이, 상기 하나 이상의 후보 경로를 산출하는 단계(S211)는, 상기 하나 이상의 함수에 대하여 오류 발생이 가능한 표현식을 식별하는 단계(S2111) 및 상기 분석 대상 프로그램에 대한 제어 흐름 그래프(CFG)를 기초로 상기 오류 발생이 가능한 표현식 위치에 도달할 수 있는 하나 이상의 후보 경로를 도출하는 단계(S2112)를 포함할 수 있다.
- [0079] 또한, 도 7에서 볼 수 있는 바와 같이, 상기 제1 오류 가능성 스코어를 도출하는 단계(S213)는, 상기 후보 경로 중 상기 오류 가능 경로의 비율을 기초로 상기 하나 이상의 함수에 대한 경로 기준 오류 가능성 스코어를 산출하는 단계(S2131) 및 상기 하나 이상의 함수에 대한 경로 기준 오류 가능성 스코어에 상기 함수에서 호출되는 피호출 함수에 대한 경로 기준 오류 가능성 스코어를 합산하여 상기 함수에 대한 제1 오류 가능성 스코어를 산출하는 단계(S2132)를 포함할 수 있다.
- [0080] 보다 구체적으로, 본 발명에서는 분석 대상 프로그램에서 널 포인트 예외(NPE) 발생 가능성이 있는 위치를 정적 분석을 통해 식별할 수 있다.
- [0081] 이와 관련하여 본 발명에 따른 정적 분석에 대한 설명을 위해 다음과 같이 분석 대상 프로그램에 대한 기호를 정의한다.
- [0082] (정의1) $S \hat{=} x = E \mid \text{return } E \mid \text{if } E \text{ } S \text{ } S \mid \text{while } E \text{ } S \mid S; S \mid \text{편}$
- [0083] (정의2) $E \hat{=} \text{new } C() \mid \text{call}(m) \mid \text{assume}(E) \mid n \mid \text{null} \mid x \mid E.E \mid \text{편}$
- [0084] 여기서, S는 문장(statement), E는 표현식(expression), x는 분석 대상 프로그램의 지역 변수 또는 멤버 변수를 의미할 수 있다.
- [0085] 보다 구체적으로, 상기 (정의1)에서는 분석 대상 프로그램의 기본 단위인 문장(statement)을 정의하는데, 구체적으로, 변수에 대한 표현식(expression) 할당, 표현식(expression) 반환, 표현식(expression)을 포함하는 if 문, while문, 복수의 문장 시퀀스(statement sequence) 등이 포함될 수 있다.
- [0086] 또한, 상기 (정의 2)에서는 표현식(expression)을 정의하는데, 구체적으로 클래스(class) 생성자(예를 들어, new C()), 함수(보다 구체적으로, 함수나 메서드 혹은 프로시저 등)의 호출, 표현식(expression) 가정, 수치(numerical value) n, null(프로그래밍 언어의 null literal), 변수(예를 들어, x), 다른 표현식(expression) 결과값의 참조(예를 들어, E.E) 등이 포함될 수 있다.
- [0087] 이에 따라, 본 발명의 일 실시예에 따른 테스트 코드 생성 장치(120)에서는, 먼저 상기 하나 이상의 함수에 대하여 오류 발생이 가능한 표현식(expression) 위치에 도달할 수 있는 하나 이상의 후보 경로를 산출할 수 있다(도 5의 path construction(320)).
- [0088] 보다 구체적으로, 테스트 코드 생성 장치(120)에서는 분석 대상 프로그램에 대한 제어 흐름 그래프(Control flow graph, CFG)를 생성할 수 있으며, 상기 제어 흐름 그래프(CFG)의 각 노드는 단위 문장(atomic statement)일 수 있다.
- [0089] 또한, 테스트 코드 생성 장치(120)에서는 상기 생성된 제어 흐름 그래프(CFG)를 활용해 널 포인트 예외(NPE) 발생 가능성이 있는 경로를 식별하기 위해 활용되는, 상기 분석 대상 프로그램의 함수(보다 구체적으로, 함수나 메서드 혹은 프로시저 등) m에 대한 타겟 표현식(expression)과 위치의 집합인 $T_{\text{exp}}(m)$ 를 도출할 수 있으며, 이때 아래 3가지 조건 중 적어도 하나를 만족해야 한다.
- [0090] (조건1) E1.E2 형태(다른 표현식(expression) 결과값의 참조)에서 참조되는 표현식(expression) E
- [0091] (조건2) 널 포인트 예외(NPE) 발생 가능성이 있는 함수 (예를 들어, 널(Null)이 발생할 수 있는 경로(Nullable Path)가 최소 1개 이상 존재하는 함수)를 호출하는 함수의 호출
- [0092] (조건3) 반환 타입이 참조(reference) 타입인 반환 변수
- [0093] 이때, 테스트 코드 생성 장치(120)에서는 분석 대상 프로그램의 각 함수 m에 대해 널 포인트 예외(NPE) 발생 가능성이 있는 표현식(expression) 집합 $T_{\text{exp}}(m)$ 을 기초로, 상기 (조건 1) 내지 (조건 3)을 충족하는 표현식이 없거나, 상기 (3)만 만족하는 경우에 대하여, 널 포인트 예외(NPE) 발생 가능성이 없는 함수로 분류될 수 있다.
- [0094] 이에 따라, 테스트 코드 생성 장치(120)에서는 표현식(expression)이 널(null)이 되거나 널(null)을 참조할 가

능성이 있는 조건에 따라 함수 m 의 $T_{exp}(m)$ 를 식별할 수 있다.

- [0095] 보다 구체적인 예를 들어, 도 8에 도시된 샘플 코드인 `addEdge` 함수 및 `getEdge` 함수에 대한 T_{exp} 를 예시해 보면 $T_{exp}(addEdge) : \{(getEdge(src), 3), (src, 4), (getEdge(target), 5)\}$ 및 $T_{exp}(getEdge) : \{(vertexMap, 2), (vertexMap, 5), (ec, 7)\}$ 가 될 수 있다.
- [0096] 이어서, 테스트 코드 생성 장치(120)에서는 상기 $T_{exp}(m)$ 에 대하여 각 표현식(즉, exp)로부터 역으로 제어 흐름 그래프(CFG)를 추적하여 경로(path, sequence of statements)를 산출할 수 있다.
- [0097] 보다 구체적인 식별 방법으로, 상기 테스트 코드 생성 장치(120)에서는 상기 표현식 위치(exp)로부터 역으로 제어 흐름 그래프(CFG)를 추적하면서, 아래 조건 중 하나 이상을 만족하는 표현식(expression)에 도달하면 경로(path) 구성을 종료할 수 있으며, 이때 아래 조건은 각 표현식(expression) 값이 널(null)이 발생할 수 있는 경로를 식별하는 조건에 대응될 수 있다.
- [0098] (조건11) 함수의 시작점
- [0099] (조건12) 다른 함수 호출
- [0100] (조건13) 표현식(exp)에 대한 부작용(side-effect)이 있는 함수 호출
- [0101] (조건14) def-use chain 상에서 표현식(exp)에서 사용(use)되는 변수에 대응되는 선언(definition)
- [0102] (조건15) 표현식(exp)과 직접 관련 있는 다른 E (expression)에 대한 $assume(E)$
- [0103] 이때, 테스트 코드 생성 장치(120)에서는 함수 m 의 $T_{exp}(m)$ 에 대하여 각각의 표현식(expression)에 대해 상기 경로(path)를 구성하고, 널 가능 경로(Nullable path) 후보로서 $set(m, Info_{path})$ 에 저장할 수 있다.
- [0104] 다음으로, 테스트 코드 생성 장치(120)에서는 상기 하나 이상의 후보 경로를 거쳐 상기 표현식에 도달하는 경우 널 발생이 가능한 오류 가능 경로를 선별할 수 있다(도 5의 Nullable path identification(330)).
- [0105] 이때, 테스트 코드 생성 장치(120)에서는 상기 널 가능 후보 경로로부터 널 포인트 예외(NPE) 발생 가능성이 존재하는 경로를 식별할 수 있다.
- [0106] 보다 구체적인 식별 방법으로서, 상기 $set(m, Info_{path})$ 의 각 후보 경로에 대해 순서대로 각 문장(statement)을 검사하는 아래 (규칙1) 내지 (규칙5)를 이용하여 식별 가능한데, 이때 아래 규칙은 각 경로 실행 후 널 포인트 예외(NPE)가 발생하게 되는 모든 경우를 포함하도록 할 수 있다.
- [0107] (규칙 1) $\Theta(x = null, isNull) = isNull[x \rightarrow true]$
- [0108] (규칙 2) $\Theta(x = y, isNull) = isNull[x \rightarrow isNull[y]]$
- [0109] (규칙 3) $\Theta(x = new C(), isNull) = isNull[x \rightarrow false]$
- [0110] (규칙 4) $\Theta(x = call(m'), isNull) = isNull[x \rightarrow ret_{null}(m')]$
- [0111] (규칙 5) $\Theta(assume(x == null), isNull) = isNull[x \rightarrow true]$
- [0112] 보다 구체적인 예를 들어, 문장(statement)이 $x = null$ 인 경우, 함수1번 규칙의 결과값은 $isNull[x \rightarrow true]$ 인데, 이는 표현식(expression) x 가 널(null)일 가능성이 있다는 의미가 될 수 있으며, 이때 $ret_{null}(m)$ 은 함수 m 이 널(null)을 반환하는 경로가 있으면 참(true), 없으면 거짓(false)을 출력할 수 있다.
- [0113] 이에 따라, 테스트 코드 생성 장치(120)에서는 상기 규칙에 기반하여 널 발생 가능 경로를 식별하는데, 경로(path)에 있는 마지막 문장(statement)까지 확인 후 결과값이 참(true)이면 해당 경로(path)는 널 포인트 예외(NPE)가 발생 가능한 경로(path)로 분류함으로써 널 가능 경로(Nullable path) 후보에 대한 $set(m, Info_{path})$ 로부터 널 가능 경로(nullable path) $mayNull(Set(m, info_{path}))$ 을 식별할 수 있다.
- [0114] 보다 구체적으로, 상기 도 8의 예를 들어, 테스트 코드 생성 장치(120)에서는 `getEdge` 함수의 $T_{exp}(getEdge)$ 로서 $expression(ec, 7)$ 을 가정하면, $path1 \rightarrow [ec = new Edge<>(edgeFactory, vertex); return ec]$, $path2 \rightarrow [!(ec == null); return ec]$ 와 같이 널 가능 경로($path1$, $path2$)를 도출하는 것이 가능하다.

- [0115] 다음으로, 테스트 코드 생성 장치(120)에서는 상기 후보 경로 및 상기 오류 가능 경로를 기초로 상기 하나 이상의 함수에 대한 제1 오류 가능성 스코어를 도출할 수 있다(도 5의 NPE-likely Score computation(340)).
- [0116] 보다 구체적으로, 테스트 코드 생성 장치(120)에서는 널 가능 경로(nullable path)를 식별한 후, 아래 수식을 기초로 각 함수에 대하여 널 포인트 예외(NPE)가 발생할 확률을 계산할 수 있다.
- [0117] (수식1) $S_{path}(m) = |mayNull(Set(m, Info_{path}))| / |Set(m, Info_{path})|$
- [0118] (수식2) $S_{NPE}(m) = S_{path}(m) + \sum_{m' \in \varphi(m)} S_{NPE}(m')$
- [0119] 이때, (수식1)에서 $mayNull(Set(m, Info_{path}))$ 은 함수 m 에서 식별한 경로 중에서 널 포인트 예외(NPE)가 발생 가능한 경로(Nullable path)의 집합을 나타내며, 즉 앞서 Path construction(도 5의 320)구성된 널 가능 경로 후보 $set(m, info_{path})$ 에 대한 Nullable path Identification(도 5의 330)을 통해 식별된 널 포인트 예외(NPE)가 발생 가능한 경로의 비율이 될 수 있다.
- [0120] 또한, (수식2)에서 정의되는 함수별 널 포인트 예외(NPE) 가능성 점수 $S_{NPE}(m)$ 는 각 함수 m 에서 널 포인트 예외(NPE)가 발생할 수 있는 확률($S_{path}(m)$) 및 함수 m 에서 호출되는 모든 함수 m' 에서 널 포인트 예외(NPE) 발생할 수 있는 확률의 합으로 산출될 수 있다.
- [0121] 즉, 함수 m 에서 재귀적으로 호출되는 모든 함수에 대한 널 포인트 예외(NPE)발생 확률의 합을 널 포인트 예외(NPE) 가능성 점수로 산출할 수 있으며, 이때 같은 함수가 여러 번 재귀적으로 호출되면 해당 함수의 점수가 한 번만 합산되도록 할 수 있다.
- [0122] 이에 따라, $S_{NPE}(m)$ 이 0인 함수의 경우 식별 대상에서 제거하는 것도 가능하다.
- [0123] 다음으로, 테스트 코드 생성 장치(120)에서는 상기 분석 대상 프로그램에 대응하는 하나 이상의 테스트 코드를 이용한 동적 분석 결과와 상기 제1 오류 가능성 스코어에 기초하여, 상기 하나 이상의 테스트 코드에 대한 제2 오류 가능성 스코어를 산출할 수 있다.
- [0124] 또한, 테스트 코드 생성 장치(120)에서는 상기 하나 이상의 테스트 코드 중에서 상기 제2 오류 가능성 스코어를 기초로 선택된 테스트 코드를 이용하여 하나 이상의 변형된 테스트 코드를 생성하는 것이 가능하다.
- [0125] 나아가, 테스트 코드 생성 장치(120)에서는 분석 대상 프로그램에 대한 테스트 코드를 무작위 및 변이를 통해 합성하여 실행하고, 실행된 테스트 중 우수한 테스트(즉 높은 코드 커버리지 달성 또는 널 포인트 예외(NPE) 탐지)를 저장하고, 이를 변이시켜 보다 개선된 테스트 코드를 만드는 과정을 반복하여 수행할 수 있다.
- [0126] 보다 구체적으로, 테스트 코드 생성 장치(120)에서는, 분석 대상 프로그램에 포함되는 각 함수 m 의 기능을 호출 및 실행하는 문장(statement)을 무작위로 조합하여 테스트 케이스를 생성할 수 있다.
- [0127] 보다 구체적으로, 테스트 코드 생성 장치(120)에서는 동적 분석을 통한 테스트 코드 실행시 호출된 함수 집합을 저장하고, 저장된 집합의 각 함수 m 에 대한 $S_{NPE}(m)$ 점수를 산출할 수 있다.
- [0128] 또한, 테스트 코드 생성 장치(120)에서는 상기 산출된 점수를 기반으로 각 테스트 코드가 널 포인트 예외(NPE)를 발생시킬 가능성을 나타내는 점수(테스트 코드별 NPE 점수)를 각 함수에 대한 점수들의 합으로 아래의 [수학식 1]과 같이 산출할 수 있다.
- [0129] 즉, 테스트 코드 생성 장치(120)에서는 함수별 널 포인트 예외(NPE) 점수뿐만 아니라 테스트 코드에 대한 널 포인트 예외(NPE) 점수를 산출하여 고려하는 것이 가능하다.
- [0130] [수학식 1]
- [0131]
$$S_{TC} = \sum_{m \in method_trace(TC)} S_{NPE}(m)$$
- [0132] 이때, 테스트 코드 생성 장치(120)에서는 상기 점수를 기반으로 테스트 코드 후보 중 해당 점수가 더 높은 테스트 케이스를 더 높은 확률로 선택할 수 있다.
- [0133] 이에 따라, 테스트 코드 생성 장치(120)에서는 선택된 테스트 코드에 대해 널 포인트 예외(NPE)를 발생시킬 가

능성이 높은 표현식(expression) 및 문장(statement)을 변이 시키는 것이 가능하다.

- [0134] 이를 위해, 테스트 코드 생성 장치(120)에서는 각 테스트 코드에 대한 정적 분석을 통해 도출된 대상 함수 m에서 널 포인트 예외(NPE)가 발생 가능한 지점에 대한 데이터 및 제어 의존성이 존재하는 변수와 문장(statement)를 식별하고 해당 변수 및 문장(statement)에 대한 변이를 더 높은 확률로 변이를 수행할 수 있다.
- [0135] 다음으로, 테스트 코드 생성 장치(120)에서는 변이된 테스트 코드 및 기존 테스트 코드의 집합으로부터 커버리지 목표(새로운 널 포인트 예외(NPE) 탐지 또는 새로운 코드 커버리지)를 달성하는 테스트 코드를 추가 선택할 수 있다.
- [0136] 이에 따라, 테스트 코드 생성 장치(120)에서는 생성된 각 테스트 코드에 대하여 동적 분석을 실행한 후, 테스트 대상 함수 별로 정적 분석에서 식별된 널 포인트 예외(NPE) 표현식(expression)을 모두 실행했다면, 발생 가능한 널 포인트 예외(NPE)를 모두 식별한 것이 되므로, 해당 함수를 테스트 대상에서 제거할 수 있다.
- [0137] 이에 따라, 테스트 코드를 통해 발견되지 않은 널 포인트 예외(NPE)가 남아있는 함수에 대해서 동적 분석에 집중하여 리소스를 절감하고 보다 효율적으로 분석을 수행하는 것이 가능하다.
- [0138] 또한, 테스트 코드 생성 장치(120)에서는 널 포인트 예외(NPE)를 탐지한 테스트 코드를 별도로 저장하여 축적할 수 있다.
- [0139] 나아가, 도 9에서는 본 발명의 일 실시예 따른 테스트 코드 생성 방법에 대한 구체적인 순서도를 예시하고 있다.
- [0140] 도 9에서 볼 수 있는 바와 같이, 테스트 코드 생성 장치(120)는 분석 대상 프로그램에 대한 정적 분석을 수행하여 널 포인트 예외(NPE) 발생 가능 함수(널 포인트 예외(NPE) 취약 영역과 밀접한 관련이 있는 함수 인수 포함) 및 코드 영역(위치)에 대응하는 $T_{exp}(m)$ 및 함수별 널 포인트 예외(NPE) 가능성 점수에 대응하는 $S_{NPE}(m)$ 을 산출할 수 있다(S310).
- [0141] 이어서, 테스트 코드 생성 장치(120)는 커버리지 목표를 계산할 수 있다(S320).
- [0142] 보다 구체적으로, 테스트 코드 생성 장치(120)는 $T_{exp}(m)$ 를 기반으로 함수별 널 포인트 예외(NPE) 발생 가능한 목표 지점을 계산하고, 해당 목표를 달성하는 테스트 코드를 생성하도록 동적 분석을 수행할 수 있다.
- [0143] 이어서, 테스트 코드 생성 장치(120)는 널 포인트 예외(NPE)가 발생하지 않은 함수를 제거할 수 있다(S330).
- [0144] 또한, 테스트 코드 생성 장치(120)는 분석 대상 프로그램에서 함수를 호출, 실행하는 문장(statement) 등을 무작위로 조합하는 방식으로 테스트 코드를 생성할 수 있다(S340).
- [0145] 다음으로, 테스트 코드 생성 장치(120)는 테스트 코드에 대한 널 포인트 예외(NPE) 점수를 산출하고, 테스트 코드의 생성, 선택 및 변이를 수행할 수 있다(S350).
- [0146] 또한, 테스트 코드 생성 장치(120)는 기존의 테스트 코드 및 변이된 테스트 코드 중 커버리지 목표 기반으로 테스트 코드를 선택할 수 있다(S360).
- [0147] 또한, 테스트 코드 생성 장치(120)는 동적 분석을 통해 테스트 대상 함수 별로 정적 분석에서 식별된 널 포인트 예외(NPE) 표현식(expression)을 모두 실행했다면, 발생 가능한 널 포인트 예외(NPE)를 모두 발견한 것이 되므로 해당 함수를 테스트 대상에서 제거할 수 있다(S370).
- [0148] 이어서, 테스트 코드 생성 장치(120)는 미리 정해진 타임아웃 시간 또는 모든 널 포인트 예외(NPE) 탐지 등 미리 정해진 종료 조건의 충족 여부를 판단하여 테스트를 종료하거나 반복 수행할 수 있다(S380).
- [0149] 이에 따라, 테스트 코드 생성 장치(120)는 다양하게 변이된 테스트 코드 및 커버리지 결과를 산출할 수 있게 된다(S390).
- [0150] 나아가, 테스트 코드 생성 장치(120)에서는 테스트를 종료한 테스트 코드에 대해서 상기 S350 내지 S370 단계를 반복하여 수행하면서, 테스트 코드에 대해 널 포인트 예외(NPE) 점수를 계산하여 테스트 코드를 선택하고 변이를 수행할 수 있다.
- [0151] 이에 따라, 테스트 코드 생성 장치(120)에서는 선택된 테스트 코드 및 변이된 테스트 코드에 대해 커버리지 목표 기반 테스트 코드를 선택하고 동적 분석을 수행한 후, 널 포인트 예외(NPE)발생 함수를 제거하는 과정을 특정 시간 또는 종료 조건을 만족할 때까지 반복하여 수행하면서, 널 포인트 예외(NPE) 발생 가능성이 높은 최적

의 테스트 코드를 축적하여 확보하는 것이 가능하게 된다.

- [0152] 또한, 본 발명의 또 다른 측면에 따른 컴퓨터 프로그램은 앞서 살핀 테스트 코드 생성 시스템(100)에서의 테스트 코드 생성 방법의 일련의 단계를 컴퓨터에서 실행시키기 위하여 컴퓨터로 판독 가능한 매체에 저장된 컴퓨터 프로그램인 것을 특징으로 한다. 상기 컴퓨터 프로그램은 컴파일러에 의해 만들어지는 기계어 코드를 포함하는 컴퓨터 프로그램뿐만 아니라, 인터프리터 등을 사용해서 컴퓨터에서 실행될 수 있는 고급 언어 코드를 포함하는 컴퓨터 프로그램일 수도 있다. 이때, 상기 컴퓨터로서는 퍼스널 컴퓨터(PC)나 노트북 컴퓨터 등에 한정되지 아니하며, 서버, 스마트폰, 태블릿 PC, PDA, 휴대전화 등 중앙처리장치(CPU)를 구비하여 컴퓨터 프로그램을 실행할 수 있는 일체의 정보처리 장치를 포함한다.
- [0153] 또한, 컴퓨터가 읽을 수 있는 매체는, 컴퓨터로 실행 가능한 프로그램을 계속 저장하거나, 실행 또는 다운로드를 위해 임시 저장하는 것일 수도 있다. 또한, 매체는 단일 또는 복수의 하드웨어가 결합된 형태의 다양한 기록수단 또는 저장수단일 수 있는데, 어떤 컴퓨터 시스템에 직접 접속되는 매체에 한정되지 않고, 네트워크 상에 분산 존재하는 것일 수도 있다. 따라서, 상기의 상세한 설명은 모든 면에서 제한적으로 해석되어서는 아니되고 예시적인 것으로 고려되어야 한다. 본 발명의 범위는 첨부된 청구항의 합리적 해석에 의해 결정되어야 하고, 본 발명의 등가적 범위 내에서의 모든 변경은 본 발명의 범위에 포함된다.
- [0154] 또한, 본 발명의 일 실시예에 따른 테스트 코드 생성 장치(120)는, 프로세서; 및 메모리를 포함하는, 테스트 코드를 생성하는 장치에 있어서, 상기 메모리는 상기 프로세서에 의해 수행될 때 상기 장치가 특정 동작을 구현하도록 구성된 명령어를 포함하고, 상기 특정 동작은: 분석 대상 프로그램에 대한 정적 분석을 통해 상기 분석 대상 프로그램에 포함되는 하나 이상의 함수에 대하여 제1 오류 가능성 스코어를 산출하는 것; 상기 분석 대상 프로그램에 대응하는 하나 이상의 테스트 코드를 이용한 동적 분석 결과와 상기 제1 오류 가능성 스코어에 기초하여, 상기 하나 이상의 테스트 코드에 대한 제2 오류 가능성 스코어를 산출하는 것; 및 상기 하나 이상의 테스트 코드 중에서 상기 제2 오류 가능성 스코어를 기초로 선택된 테스트 코드를 이용하여 하나 이상의 변형된 테스트 코드를 생성하는 것을 포함할 수 있다.
- [0155] 이때, 본 발명의 일 실시예에 따른 테스트 코드 생성 장치(120)는 앞서 도 1 내지 도 9를 참조하여 설명한 테스트 코드 생성 방법, 장치 및 시스템에 대한 설명을 기초로 용이하게 구현하여 실시할 수 있는 바, 아래에서는 반복된 설명을 생략하고 본 발명의 주요 구성을 중심으로 기술한다.
- [0156] 여기서, 상기 제1 오류 가능성 스코어를 산출하는 것에는, 상기 하나 이상의 함수에 대하여 오류 발생이 가능한 표현식(expression) 위치에 도달할 수 있는 하나 이상의 후보 경로를 산출하는 것; 상기 하나 이상의 후보 경로를 거쳐 상기 표현식에 도달하는 경우 오류 발생이 가능한 오류 가능 경로를 선별하는 것; 및 상기 후보 경로 및 상기 오류 가능 경로를 기초로 상기 하나 이상의 함수에 대한 제1 오류 가능성 스코어를 도출하는 것이 포함될 수 있다.
- [0157] 또한, 상기 하나 이상의 후보 경로를 산출하는 것에는, 상기 하나 이상의 함수에 대하여 오류 발생이 가능한 표현식을 식별하는 것; 및 상기 분석 대상 프로그램에 대한 제어 흐름 그래프(CFG)를 기초로 상기 오류 발생이 가능한 표현식 위치에 도달할 수 있는 하나 이상의 후보 경로를 도출하는 것이 포함될 수 있다.
- [0158] 또한, 상기 하나 이상의 후보 경로를 도출하는 것에서는, 상기 오류 발생이 가능한 표현식 위치를 기준으로 상기 제어 흐름 그래프를 역방향으로 추적하여 상기 하나 이상의 후보 경로를 도출할 수 있다.
- [0159] 또한, 상기 오류 가능 경로를 선별하는 것에서는, 상기 하나 이상의 후보 경로를 따라 각 문장(statement)을 실행하면서 오류 발생 여부를 판단하여 오류 가능 경로를 선별할 수 있다.
- [0160] 또한, 상기 제1 오류 가능성 스코어를 도출하는 것에는, 상기 후보 경로 중 상기 오류 가능 경로의 비율을 기초로 상기 하나 이상의 함수에 대한 경로 기준 오류 가능성 스코어를 산출하는 것; 및 상기 하나 이상의 함수에 대한 경로 기준 오류 가능성 스코어에 상기 함수에서 호출되는 피호출 함수에 대한 경로 기준 오류 가능성 스코어를 합산하여 상기 함수에 대한 제1 오류 가능성 스코어를 산출하는 것이 포함될 수 있다.
- [0161] 이때, 상기 제2 오류 가능성 스코어를 산출하는 것에서는, 하나 이상의 테스트 코드를 이용한 동적 분석에서 호출된 하나 이상의 함수에 대한 각 제1 오류 가능성 스코어를 합산하여 상기 제2 오류 가능성 스코어를 산출할 수 있다.
- [0162] 또한, 상기 변형된 테스트 코드를 생성하는 것에서는, 상기 하나 이상의 테스트 코드에 대한 정적 분석을 통해 오류 발생과 연관된 변수 또는 문장을 식별하고, 식별된 변수 또는 문장을 확률 기반으로 선택하여 변형된 테스트

트 코드를 생성할 수 있다.

- [0163] 또한, 상기 변형된 테스트 코드를 생성하는 것에서는, 상기 변형된 테스트 코드를 기초로 동적 분석을 반복 수행하면서, 상기 제2 오류 가능성 스코어를 기초로 변형된 테스트 코드를 반복 생성할 수 있다.
- [0164] 또한, 도 10에서는 본 발명의 제안 방법이 적용될 수 있는 장치(50)를 예시한다.
- [0165] 도 10을 참조하면, 상기 장치(50)는 본 발명의 제안 방법에 따른 테스트 코드 생성 시스템(100)에서의 테스트 코드 생성 프로세스를 구현하도록 구성될 수 있다.
- [0166] 예를 들어, 본 발명의 제안 방법이 적용될 수 있는 장치(50)는 리피터, 허브, 브리지, 스위치, 라우터, 게이트웨이 등과 같은 네트워크 장치, 데스크톱 컴퓨터, 워크스테이션 등과 같은 컴퓨터 장치, 스마트폰 등과 같은 이동 단말, 랩톱 컴퓨터 등과 같은 휴대용 기기, 디지털 TV 등과 같은 가전 제품, 자동차 등과 같은 이동 수단 등을 포함할 수 있다. 다른 예로, 본 발명이 적용될 수 있는 장치(50)는 SoC(System On Chip) 형태로 구현된 ASIC(Application Specific Integrated Circuit)의 일부로 포함될 수 있다.
- [0167] 메모리(20)는 프로세서(10)와 동작 시 연결될 수 있고, 프로세서(10)의 처리 및 제어를 위한 프로그램 및/또는 명령어들을 저장할 수 있고, 본 발명에서 사용되는 데이터와 정보, 본 발명에 따른 데이터 및 정보 처리를 위해 필요한 제어 정보, 데이터 및 정보 처리 과정에서 발생하는 임시 데이터 등을 저장할 수 있다. 메모리(20)는 ROM(Read Only Memory), RAM(Random Access Memory), EPROM(Erasable Programmable Read Only Memory), EEPROM(Electrically Erasable Programmable Read-Only Memory), 플래쉬(flash) 메모리, SRAM(Static RAM), HDD(Hard Disk Drive), SSD(Solid State Drive) 등과 같은 저장 장치로서 구현될 수 있다.
- [0168] 프로세서(10)는 메모리(20) 및/또는 네트워크 인터페이스(30)와 동작 시 연결(operatively connected)될 수 있고, 장치(50) 내 각 모듈의 동작을 제어한다. 특히, 프로세서(10)는 본 발명의 제안 방법을 수행하기 위한 각종 제어 기능을 수행할 수 있다. 프로세서(10)는 컨트롤러(controller), 마이크로 컨트롤러(microcontroller), 마이크로 프로세서(microprocessor), 마이크로 컴퓨터(microcomputer) 등으로도 불릴 수 있다. 본 발명의 제안 방법은 하드웨어(hardware) 또는 펌웨어(firmware), 소프트웨어, 또는 이들의 결합에 의해 구현될 수 있다. 하드웨어를 이용하여 본 발명을 구현하는 경우에는, 본 발명을 수행하도록 구성된 ASIC(application specific integrated circuit) 또는 DSP(digital signal processor), DSPD(digital signal processing device), PLD(programmable logic device), FPGA(field programmable gate array) 등이 프로세서(10)에 구비될 수 있다. 한편, 펌웨어나 소프트웨어를 이용하여 본 발명의 제안 방법을 구현하는 경우에는 펌웨어나 소프트웨어는 본 발명의 제안 방법을 구현하는 데 필요한 기능 또는 동작들을 수행하는 모듈, 절차 또는 함수 등과 관련된 명령어(instruction)들을 포함할 수 있으며, 명령어들은 메모리(20)에 저장되거나 메모리(20)와 별도로 컴퓨터 판독가능한 기록 매체(미도시)에 저장되어 프로세서(10)에 의해 실행될 때 장치(50)가 본 발명의 제안 방법을 구현하도록 구성될 수 있다.
- [0169] 또한, 장치(50)는 네트워크 인터페이스 디바이스(network interface device)(30)를 포함할 수 있다. 네트워크 인터페이스 디바이스(30)는 프로세서(10)와 동작 시 연결되며, 프로세서(10)는 네트워크 인터페이스 디바이스(30)를 제어하여 무선/유선 네트워크를 통해 정보 및/또는 데이터, 신호, 메시지 등을 나르는 무선/유선 신호를 전송 또는 수신할 수 있다. 네트워크 인터페이스 디바이스(30)는 예를 들어 IEEE 802 계열, 3GPP LTE(-A), 3GPP 5G 등과 같은 다양한 통신 규격을 지원하며, 해당 통신 규격에 따라 제어 정보 및/또는 데이터 신호를 송수신할 수 있다. 네트워크 인터페이스 디바이스(30)는 필요에 따라 장치(50) 밖에 구현될 수도 있다.
- [0170] 이에 따라, 본 발명의 일 실시예에 따른 복합 분석을 활용한 테스트 코드 생성 방법, 장치, 시스템 및 컴퓨터 프로그램에서는, 분석 대상 프로그램에 대한 취약점을 탐지하기 위한 테스트 코드를 효율적으로 생성할 수 있고, 또한 코드 커버리지가 증가하더라도 널 포인트 예외(NPE) 등의 오류에 대한 취약점 탐지 능력이 정제되거나 저하되는 문제를 개선할 수 있으며, 나아가 정적 분석과 동적 분석을 동시에 활용하여 분석 대상 프로그램의 취약점을 효율적으로 탐지할 수 있게 된다.
- [0171] 본 명세서에서 설명된 위 실시예 및 도면들은 단지 예시적인 것일 뿐, 어떠한 방법으로도 본 발명의 범위를 한정하는 것은 아니다. 또한, 도면에 도시된 구성요소들 간의 선들의 연결 또는 연결 부재들은 기능적인 연결 및/또는 물리적 또는 회로적 연결들을 예시적으로 나타낸 것으로서, 실제장치에서는 대체 가능하거나 추가의 다양한 기능적인 연결, 물리적인 연결, 또는 회로 연결들로서 나타내어질 수 있다. 또한, "필수적인", "중요하게" 등과 같이 구체적인 언급이 없다면 본 발명의 적용을 위하여 반드시 필요한 구성요소가 아닐 수 있다.
- [0172] 본 발명의 명세서(특히 특허청구범위에서)에서 "상기"의 용어 및 이와 유사한 지시 용어의 사용은 단수 및 복수

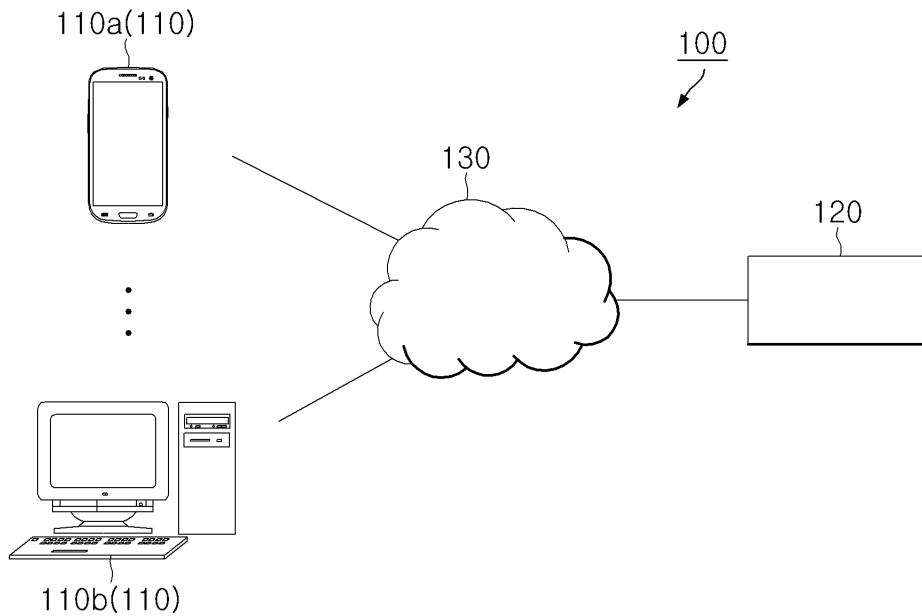
모두에 해당하는 것일 수 있다. 또한, 본 발명에서 범위(range)를 기재한 경우 상기 범위에 속하는 개별적인 값을 적용한 발명을 포함하는 것으로서(이에 반하는 기재가 없다면), 발명의 상세한 설명에 상기 범위를 구성하는 각 개별적인 값을 기재한 것과 같다. 또한 본 발명 중 방법 발명에서 제시하는 단계들은 반드시 그 선후의 순서에 대한 구속을 의도한 것이 아니며, 각 공정의 본질에 따라 반드시 어느 단계가 선행되어야 하는 것이 아닌 한 순서는 필요에 따라 적절히 변경될 수 있다. 본 발명에서 모든 예들 또는 예시적인 용어(예들 들어, 등등)의 사용은 단순히 본 발명을 상세히 설명하기 위한 것으로서 특허청구범위에 의해 한정되지 않는 이상 상기 예들 또는 예시적인 용어로 인해 본 발명의 범위가 한정되는 것은 아니다. 또한, 통상의 기술자는 다양한 수정, 조합 및 변경이 부가된 특허청구범위 또는 그 균등 범주 내에서 설계 조건 및 요소에 따라 구성될 수 있음을 이해할 수 있다.

부호의 설명

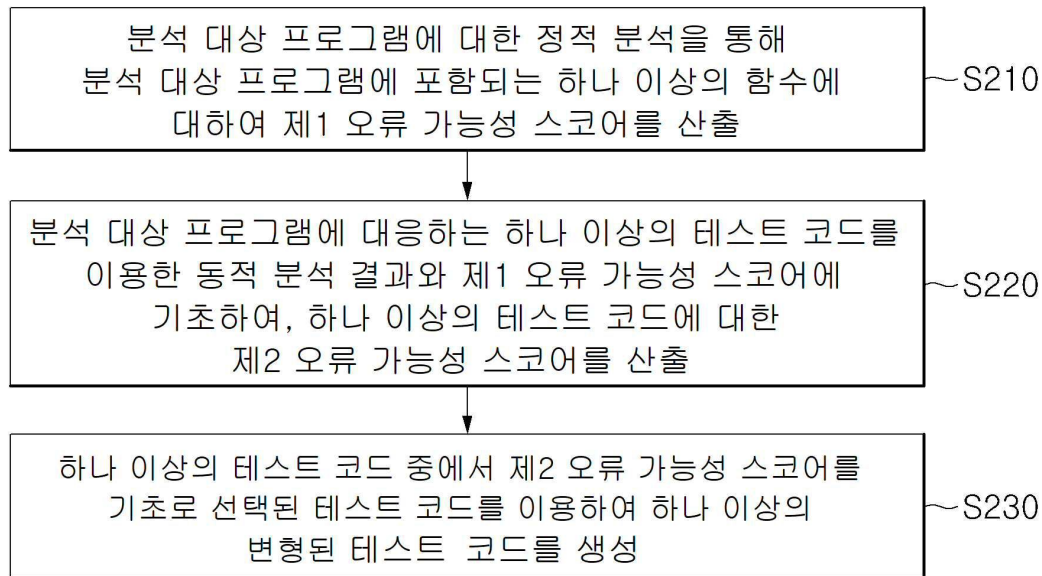
- [0173]
- 10 : 프로세서
 - 20 : 메모리
 - 30 : 인터페이스 디바이스
 - 50 : 장치
 - 100 : 테스트 코드 생성 시스템
 - 110a, 110b : 단말
 - 120 : 테스트 코드 생성 장치
 - 130 : 네트워크

도면

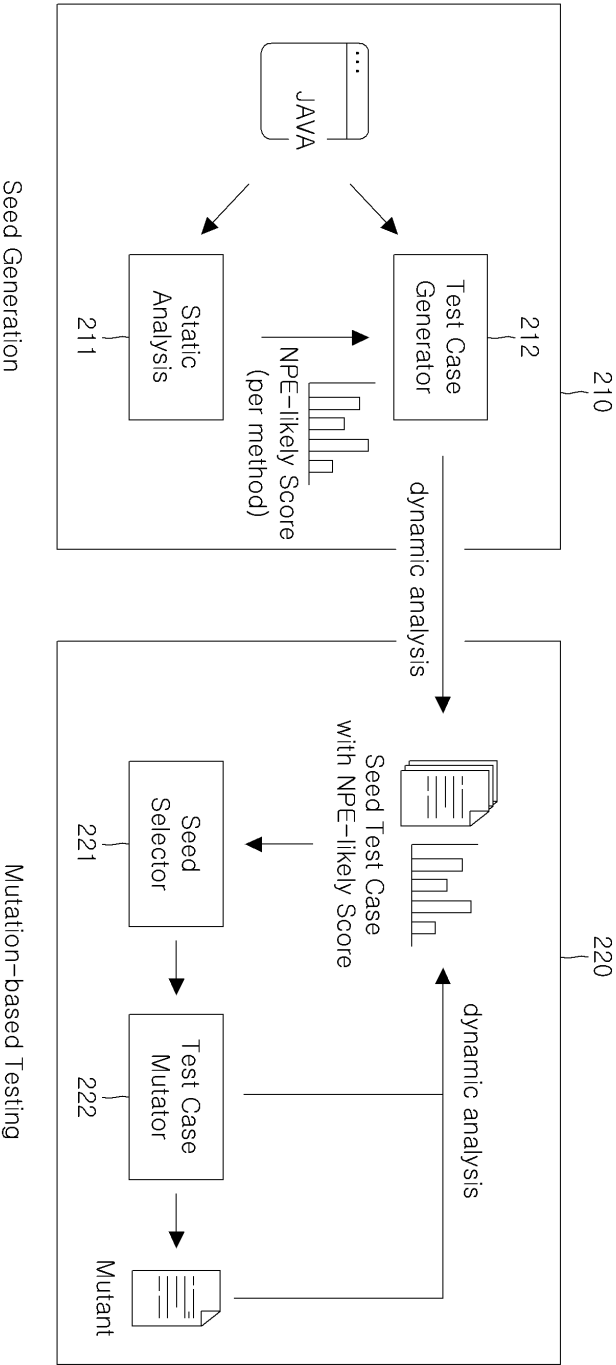
도면1



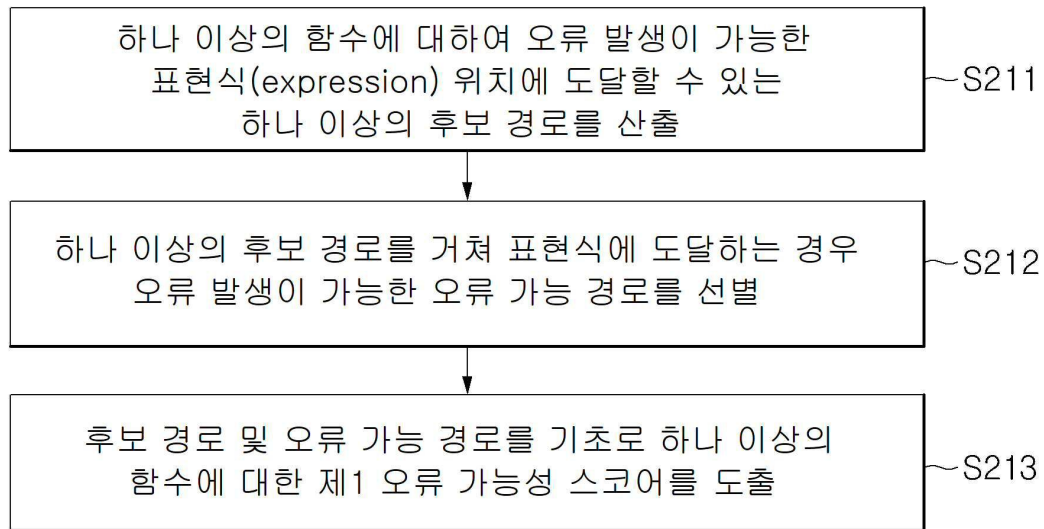
도면2



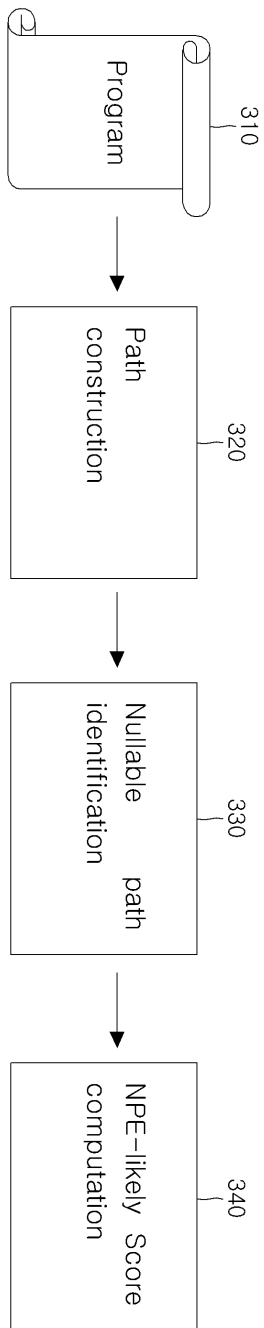
도면3



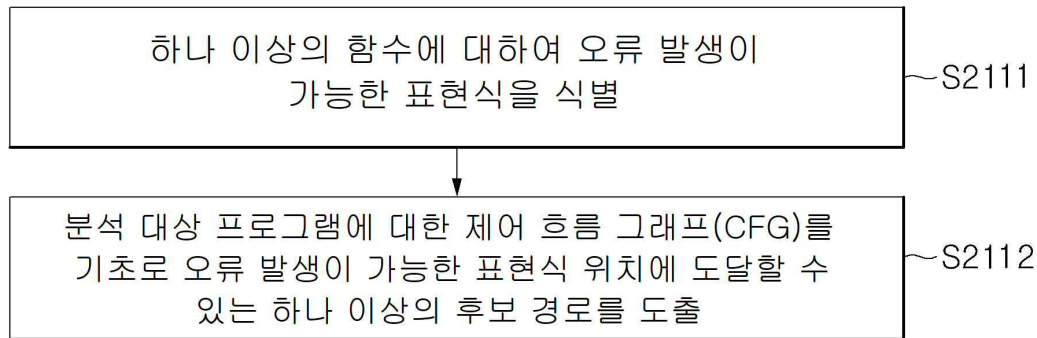
도면4



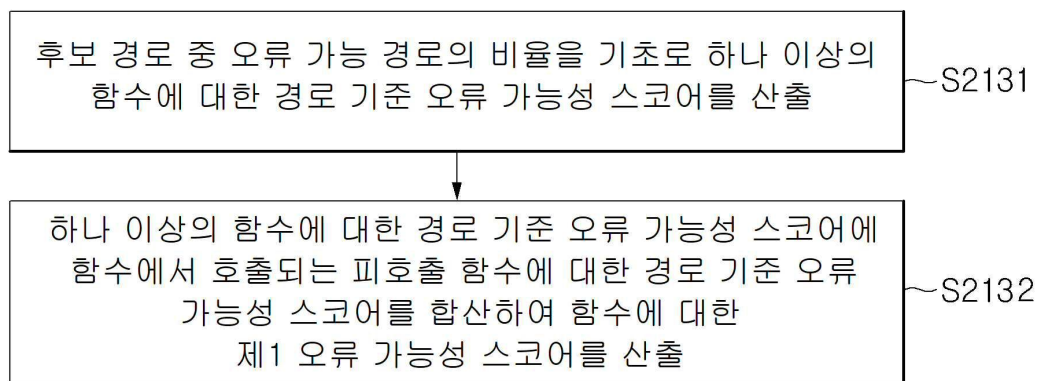
도면5



도면6



도면7



도면8

```

1  public boolean addEdge(V src, V target) {
2      if (src == null) return false;
3      getEdge(src).addEdge(e);
4      if (!src.equals(target)) {
5          if getEdge(target).addEdge(e);
6      }
7      return true;
8  }

```

(a) addEdge method

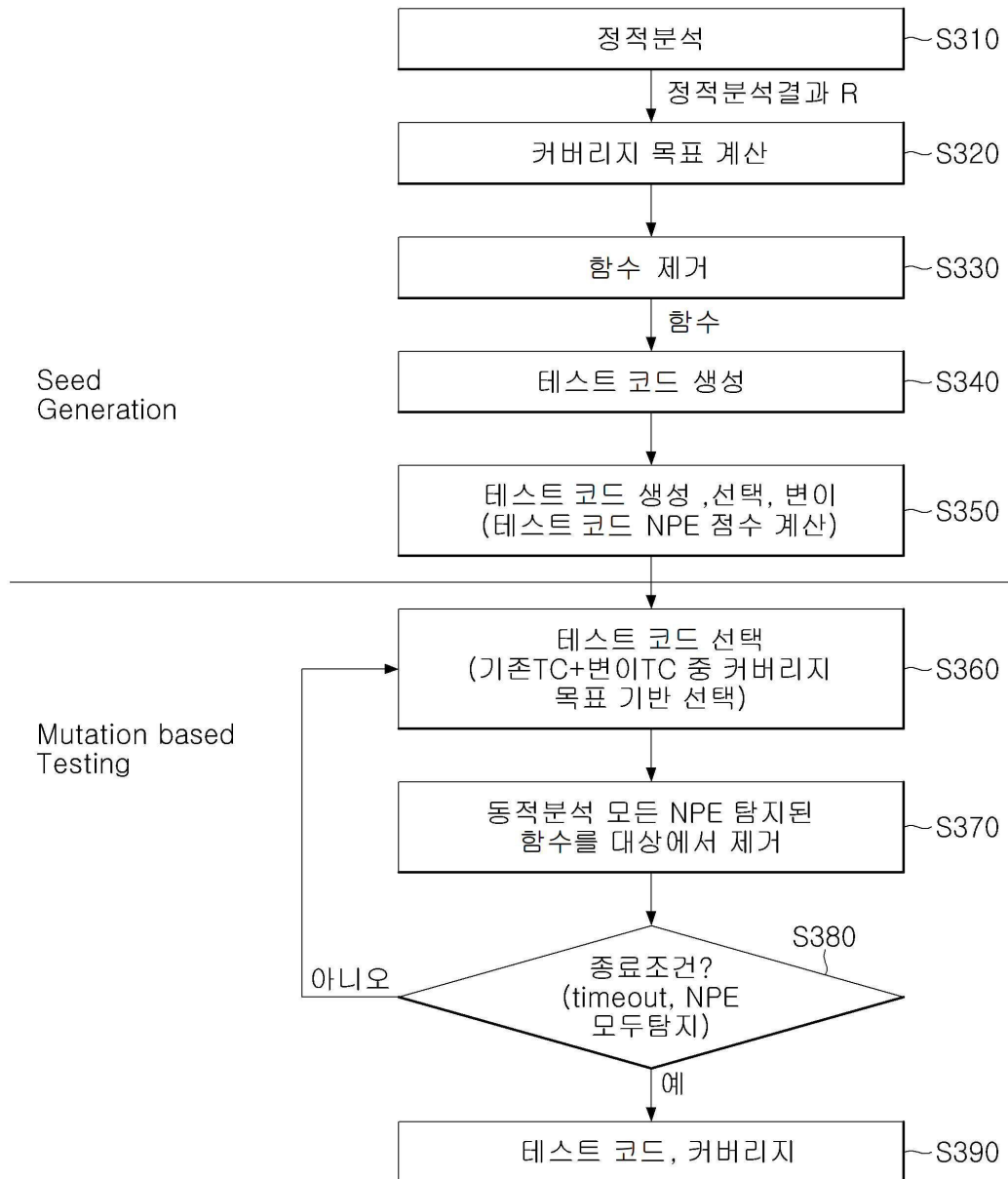
```

1  public Edge<V, E> getEdge(V vertex) {
2      Edge<V, E> ec = vertexMap.get(vertex);
3      if (ec == null) {
4          ec = new Edge<>(edgeFactory, vertex);
5          vertexMap.put(vertex, ec);
6      }
7      return ec;
8  }

```

(b) getEdge method

도면9



도면10

