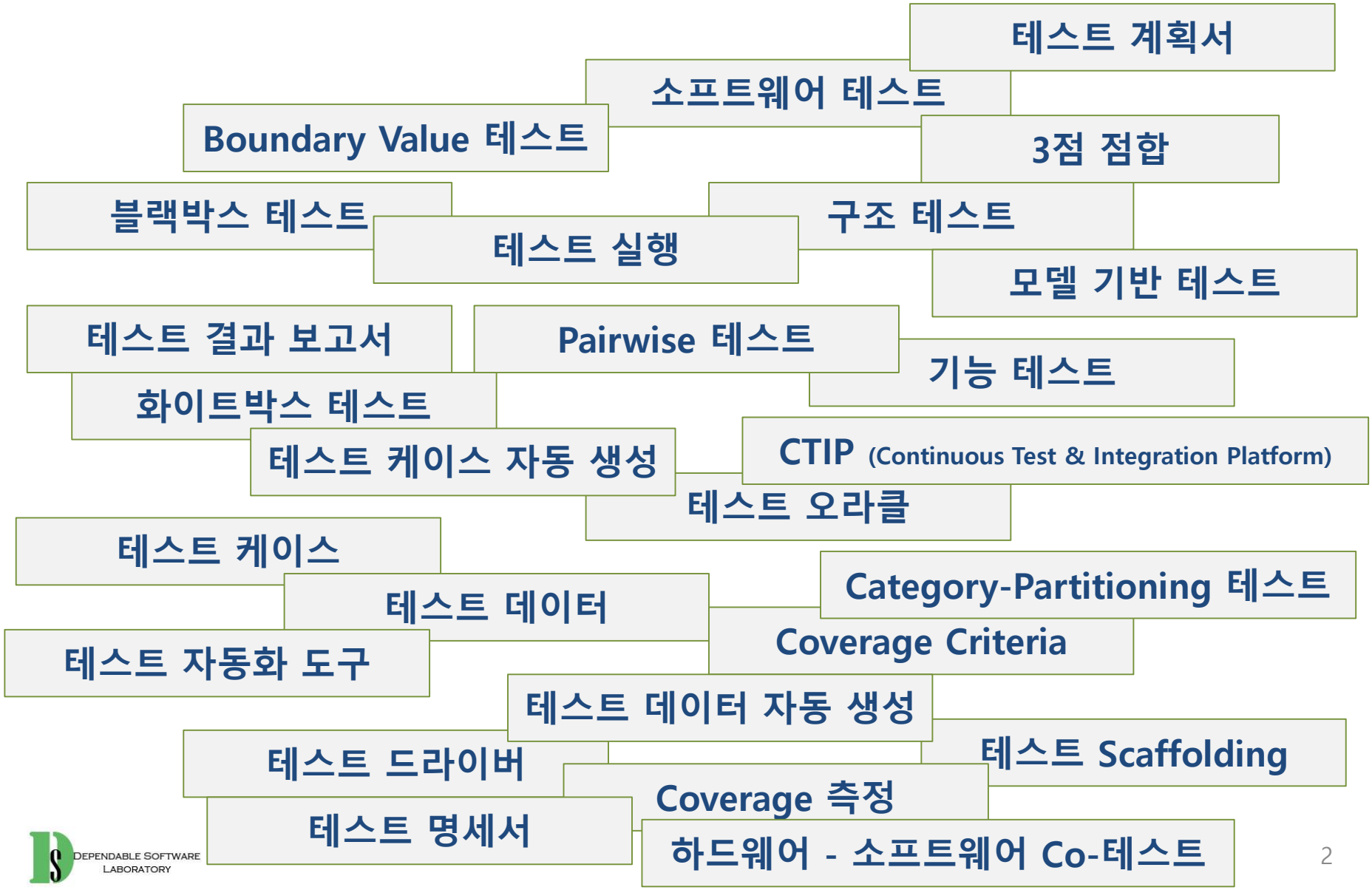
소프트웨어 테스트

- 개념, 기법 및 활용 -

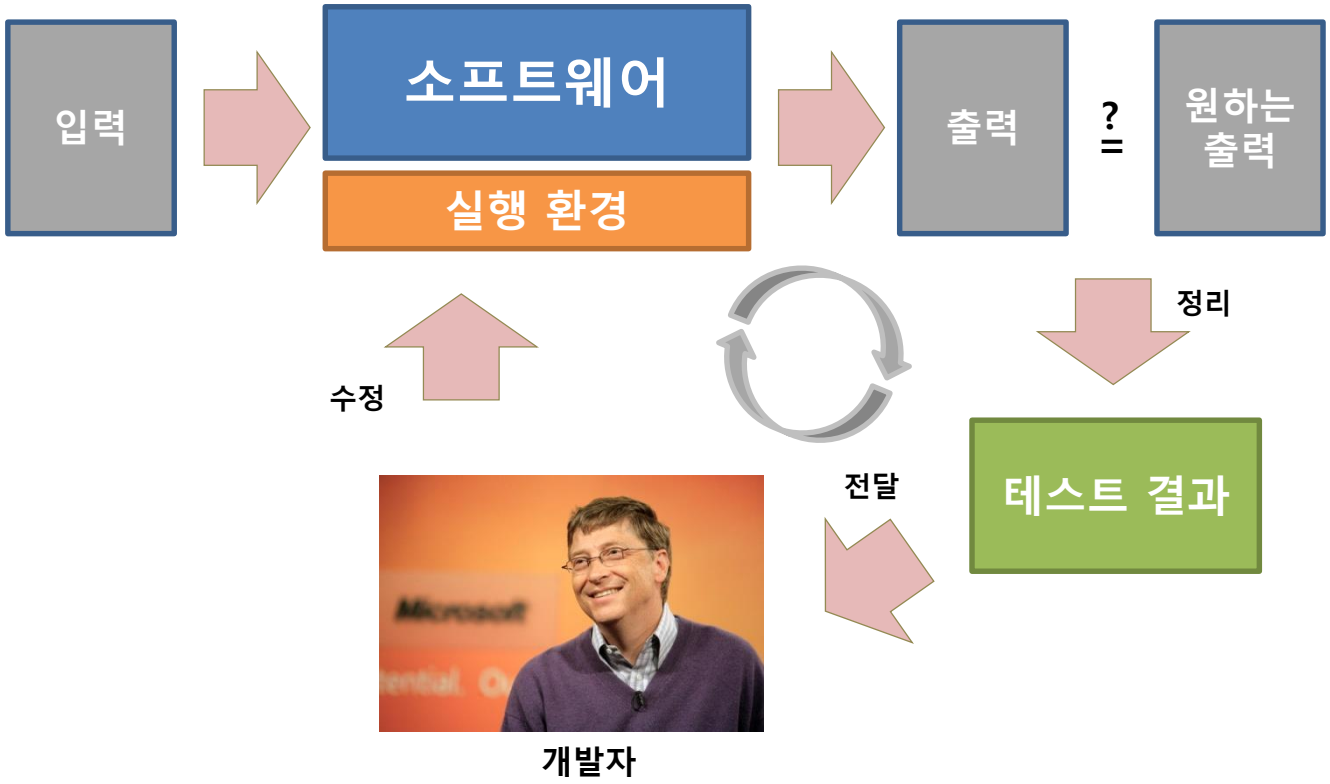
Dependable Software Laboratory
정세진

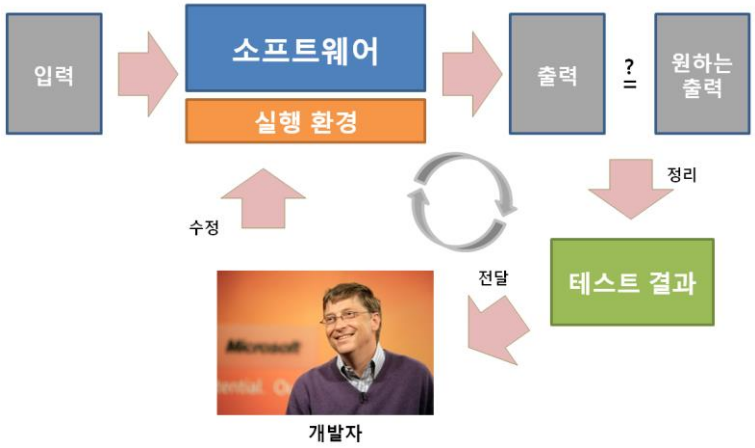
테스트 분야 개념들


테스트 계획서
소프트웨어 테스트
Boundary Value 테스트
3점 점합
블랙박스 테스트
구조 테스트
테스트 실행
모델 기반 테스트
테스트 결과 보고서
Pairwise 테스트
기능 테스트
화이트박스 테스트
테스트 케이스 자동 생성
CTIP (Continuous Test & Integration Platform)
테스트 오라클
테스트 케이스
Category-Partitioning 테스트
테스트 데이터
Coverage Criteria
테스트 자동화 도구
테스트 데이터 자동 생성
테스트 Scaffolding
테스트 드라이버
Coverage 측정
테스트 명세서
하드웨어 - 소프트웨어 Co-테스트

소프트웨어 테스트

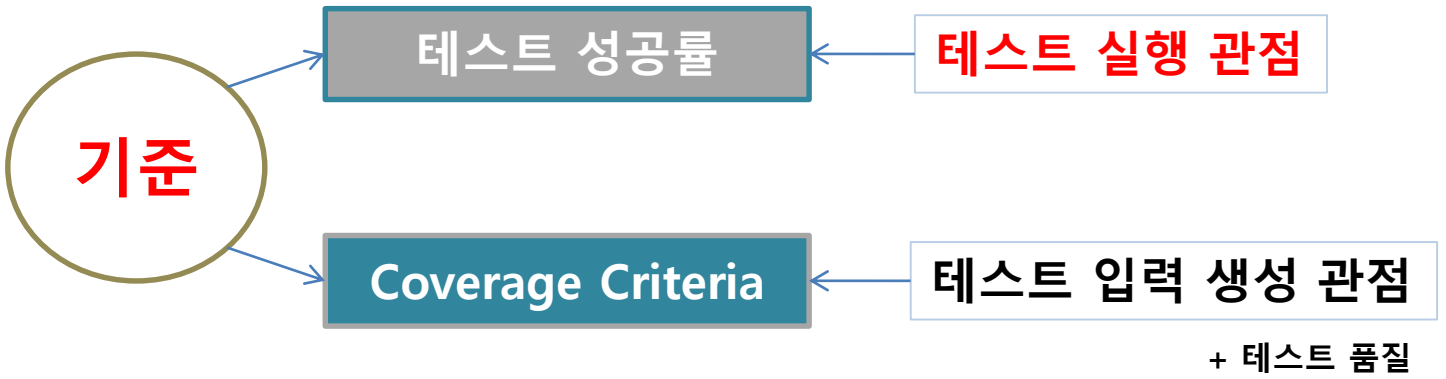
입력을 주고 소프트웨어를 직접 실행함으로써, 원하는 출력이 생성되는지 확인하는 일련의 작업





언제까지 반복?

기준을 만족할 때 까지



블랙박스 vs. 화이트박스

블랙박스 테스트

기능 테스트 (Functional Test)

있어야 할 기능이 정확하게 구현되어 있는가?
기능명세서, 요구사항명세서, 사용자설명서
모든 기능을 테스트 입력으로 만들어야
내부 구현은 관심 無

화이트박스 테스트

구조 테스트 (Structural Test)

구현 프로그램의 가능한 모든 경우(Execution Path)를 시험해 봤는가?
CFG (Control Flow Graph) , DFG (Data Flow Graph)
특정 Coverage Criteria를 만족하는 테스트 입력을 만들어야
기능의 구현 여부는 관심 無

1. 시스템시험(System Test)은 어떤 종류의 테스트입니까?

블랙박스 테스트

2. 단위시험(Unit Test)은 어떤 종류의 테스트입니까?

두 종류 모두 가능

블랙박스 테스트

블랙박스 테스트

명세서에 정의된 기능이 잘 구현되어 있는지 확인하는 테스트

가장 기본이 되는 테스트 (A Base-line Test)

= 기능 테스트 (Functional Test)

= 명세 기반 테스트 (Specification-based Test)

= 체계적 분할 기반 테스트 (Systematic Partitioning-based Test)

기본 이론 : **체계적 분할 (Systematic Partitioning)**

체계적 분할

기본 이론 : **체계적 분할 (Systematic Partitioning)**

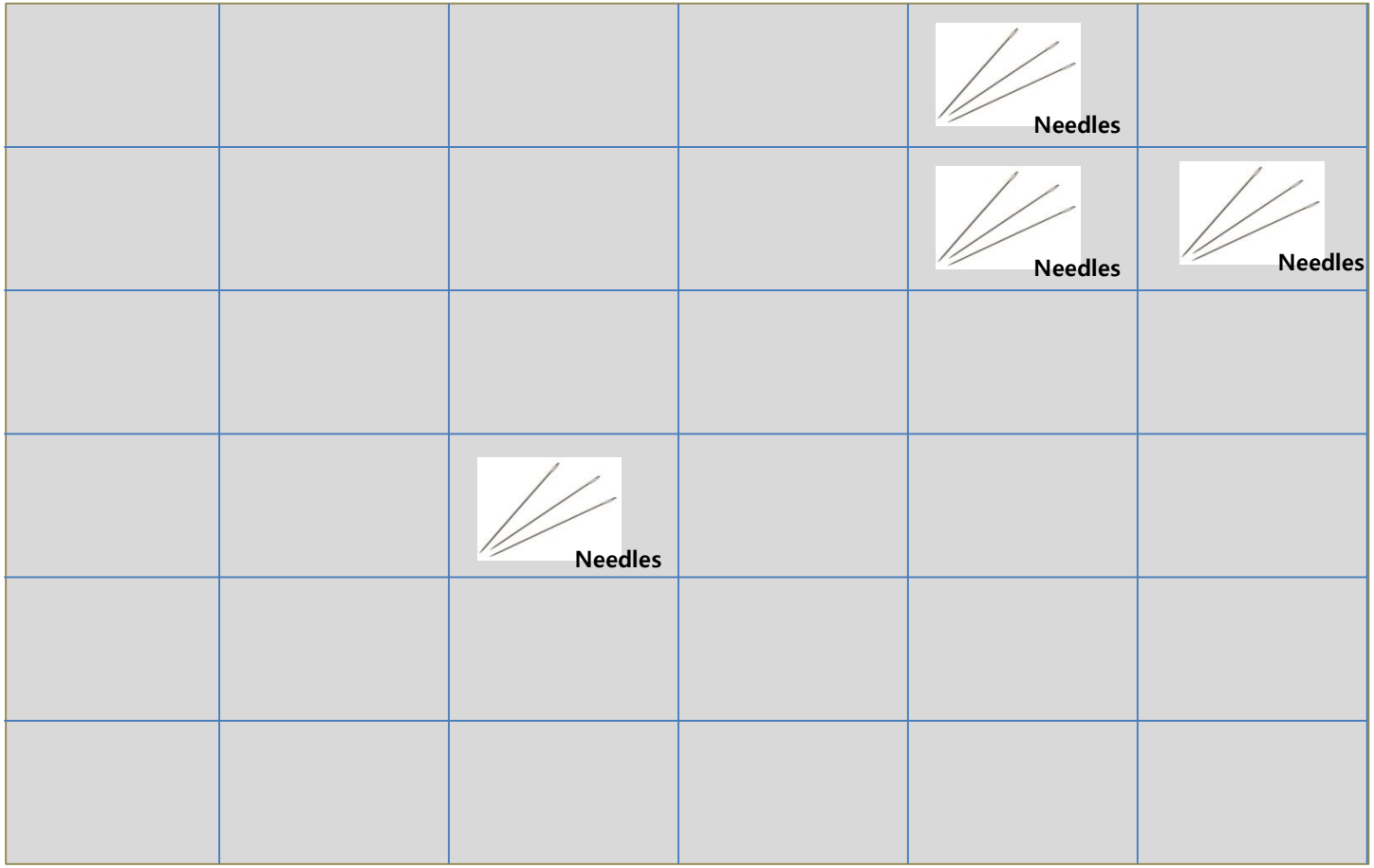
특정 정보를 이용하여, 전체 테스트 입력 범위 중 오류를 잘 찾아낼 수 있는 입력구간(region of input space)을 찾아낸다.

∵ 전체 입력 범위를 모두 테스트 할 수 없다.

∵ 오류들은 주로 몰려있다.

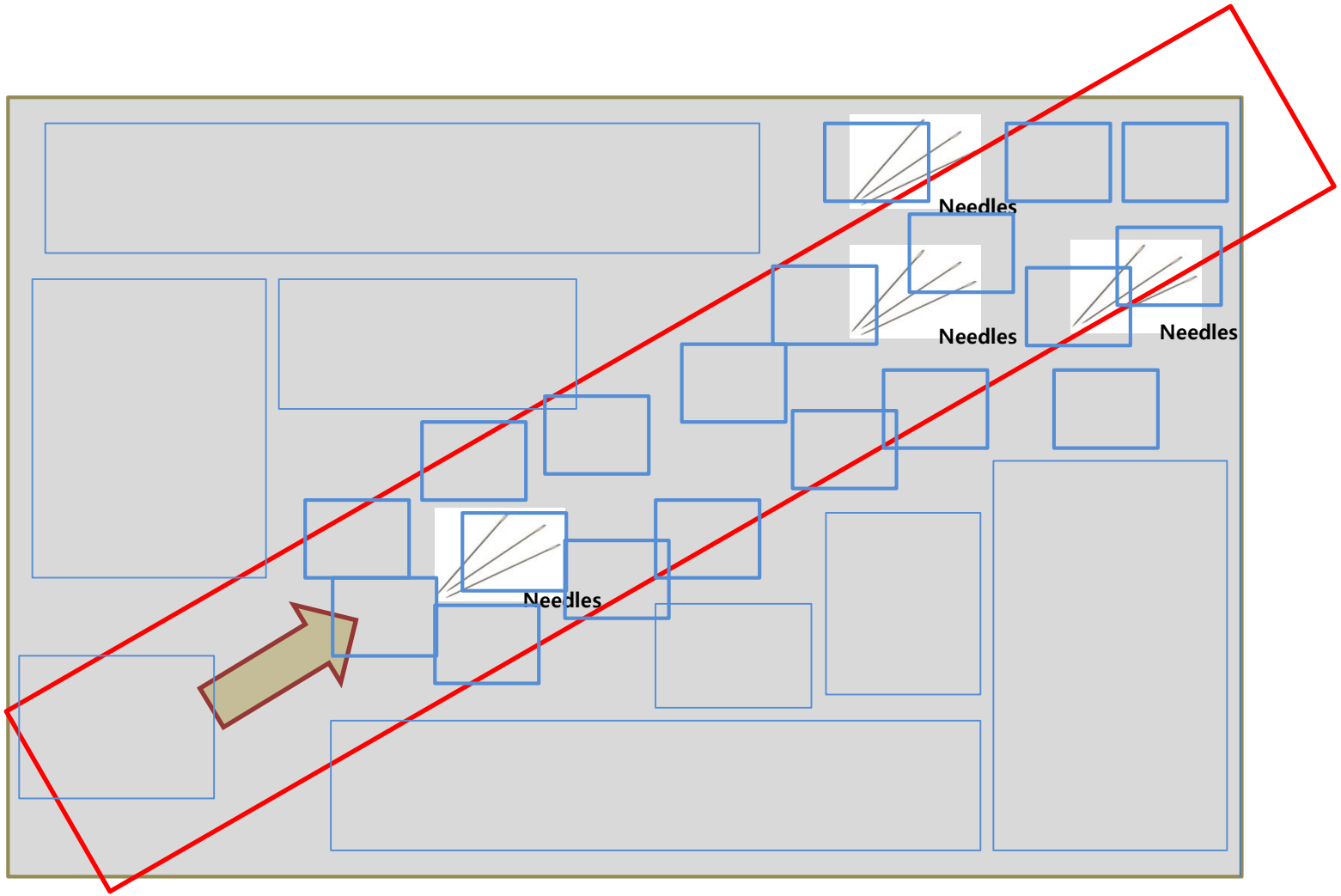
균등 분할

= Random Testing
= Uniform Testing



체계적 분할

동일한 Cost로 수행되는 테스트 범위



테스트 비용(회수): 27회
테스트 효율: 高

정보는 어디에?

명세서 (Specification)

- 프로젝트 계획서 (Project Plan)
- 요구사항 명세서 (Requirements Specification)
- 디자인 명세서 (Design Specification)
- 테스트 계획서 (Test Plan)
- 각종 모든 문서

기능 테스트

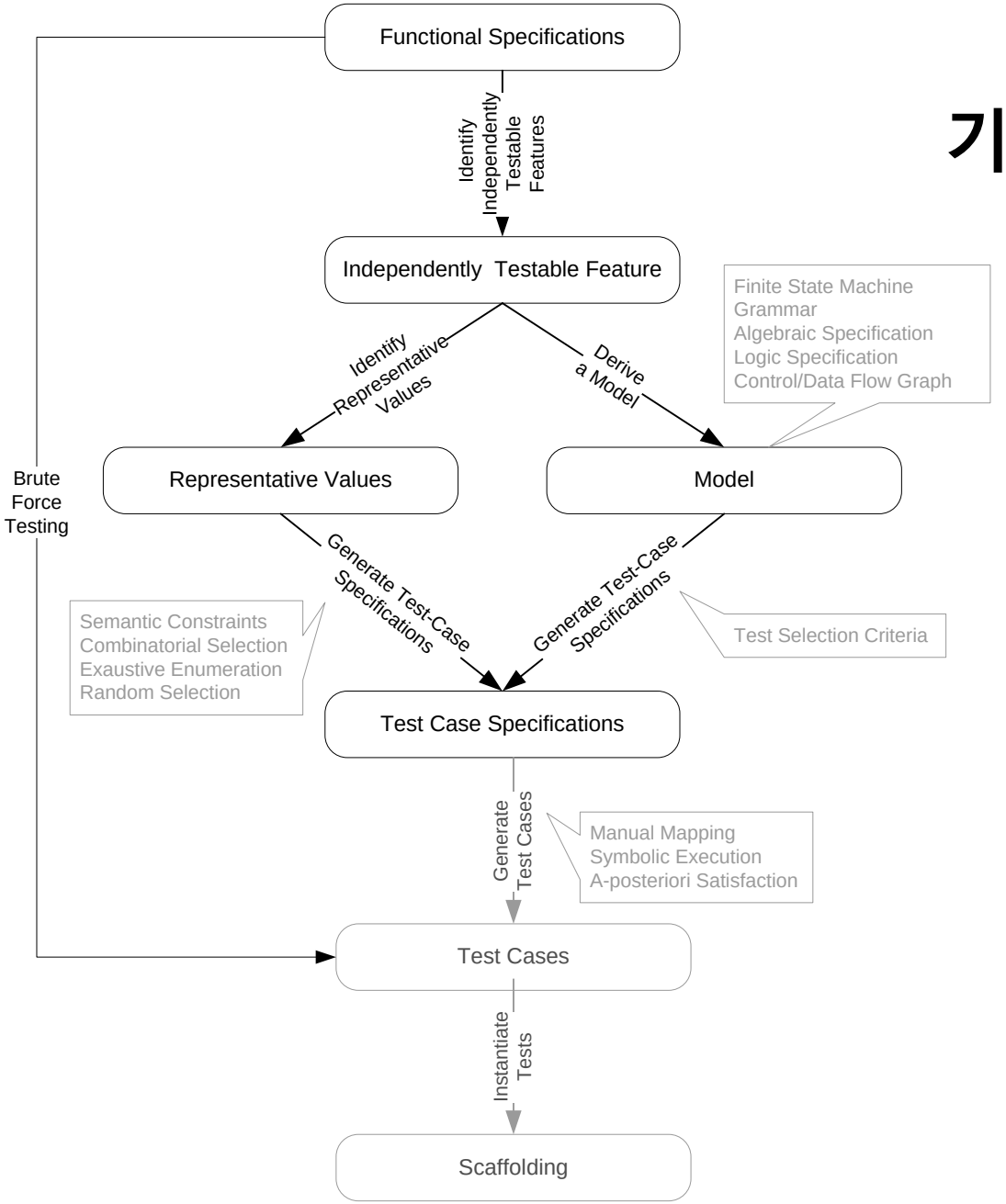
기능 테스트 ≈ 블랙박스 테스트 ≈ 명세 기반 테스트

→ 체계적 분할 기반 테스트

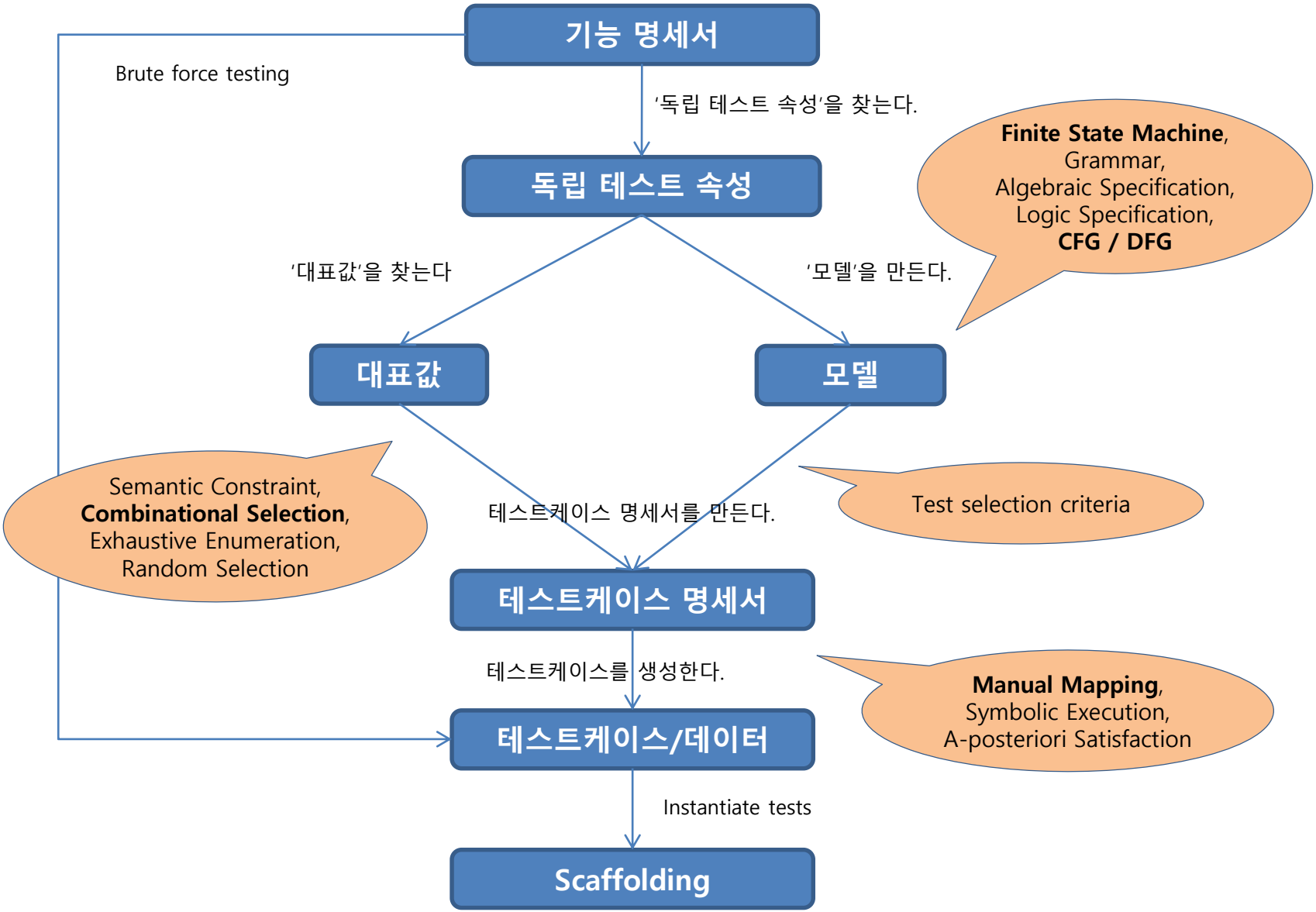
명세(Specification)를 사용하여 입력 범위를 분할(partitioning of input space)

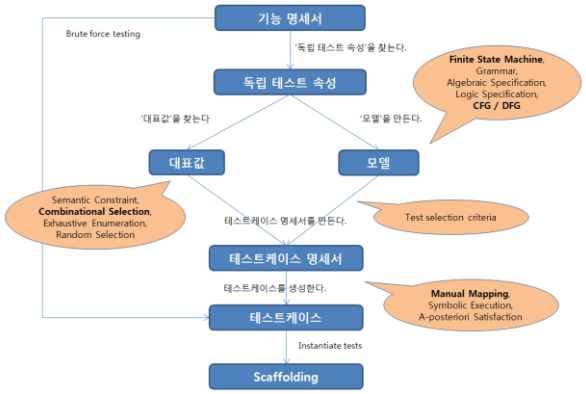
각 분할된 범위(Category)를 테스트

기능 테스트 단계



Source: "Software Testing and Analysis" by Mauro Pezzè and Michal Young, WILEY


기능 명세서
Brute force testing
'독립 테스트 속성'을 찾는다.
독립 테스트 속성
Finite State Machine,
Grammar,
Algebraic Specification,
Logic Specification,
CFG / DFG
'대표값'을 찾는다
'모델'을 만든다.
대표값
모델
Semantic Constraint,
Combinational Selection,
Exhaustive Enumeration,
Random Selection
테스트케이스 명세서를 만든다.
Test selection criteria
테스트케이스 명세서
테스트케이스를 생성한다.
Manual Mapping,
Symbolic Execution,
A-posteriori Satisfaction
테스트케이스/데이터
Instantiate tests
Scaffolding

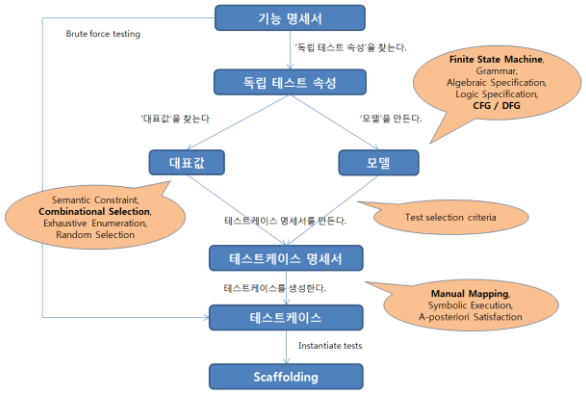


기능명세서 분석

전통적인
블랙박스 테스트

모델 기반 테스트

테스트 데이터 생성



수동

기능명세서 분석

현실은?

전통적인
블랙박스 테스트

수동

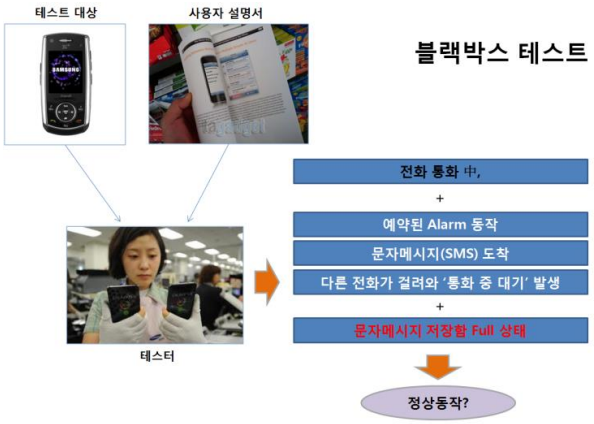
모델 기반 테스트

Brute Force Test
→ 막 테스트

-> 개발자가 수행하는 brute force test는 기대 이상의 성능을 보임

수동

테스트 데이터 생성



기능 테스트를 잘 하기 위해서는?

1. 기능명세서를 꿰자.

2. 기능 테스트의 기본에 충실하자.

Systematic Partitioning-based Test

오류가 많을 만한 곳을 중점적으로 공략

3. 사람 중심으로 운영하자.

손으로 다 할 수 있어야 도구도 잘 쓴다.

대표적인 블랙박스 테스트 기법들

Category-Partitioning Test

1. '독립 테스트 속성'을 찾는다.
2. 각 속성별로 '대표값'을 찾는다.
3. 테스트케이스를 생성한다.

Pairwise Test

1. Category-Partitioning Test를 수행한다.
2. 관련 있는 속성들을 2~4 단위로 묶는다.
3. 자동화 도구를 활용한다.

Catalog-based Test

1. 오래 동안 유사한 테스트를 수행한다.
2. 대표적인 속성을 Catalog로 만든다.
3. 위 테스트들을 수행한다.

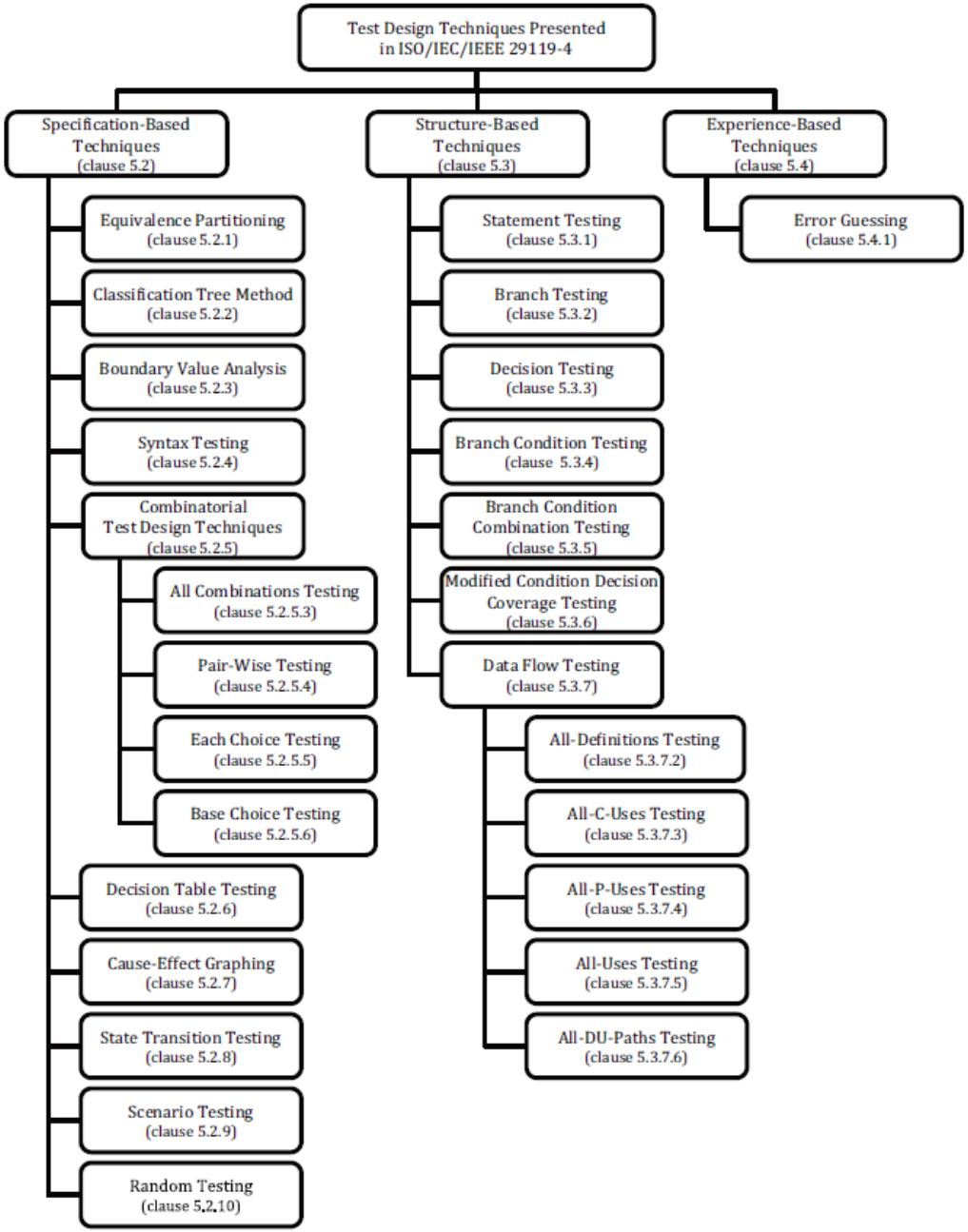


Figure 2 — The set of test design techniques presented in ISO/IEC/IEEE 29119-4

Category-Partitioning Test

예제: 모니터 디스플레이 테스트

Step 1. '독립 테스트 속성'을 찾는다.

Display Mode	Language	Fonts	Color	Screen Size

Step 2. '대표값'을 찾는다.

Display Mode	Language	Fonts	Color	Screen Size
Full-Graphics	English	Minimal	Monochrome	Hand-Held
Text-Only	French	Standard	Color-Map	Laptop
Limited-Bandwidth	Spanish	Document-Embedded	16-Bits	Full-Size
	Korean		True-Color	

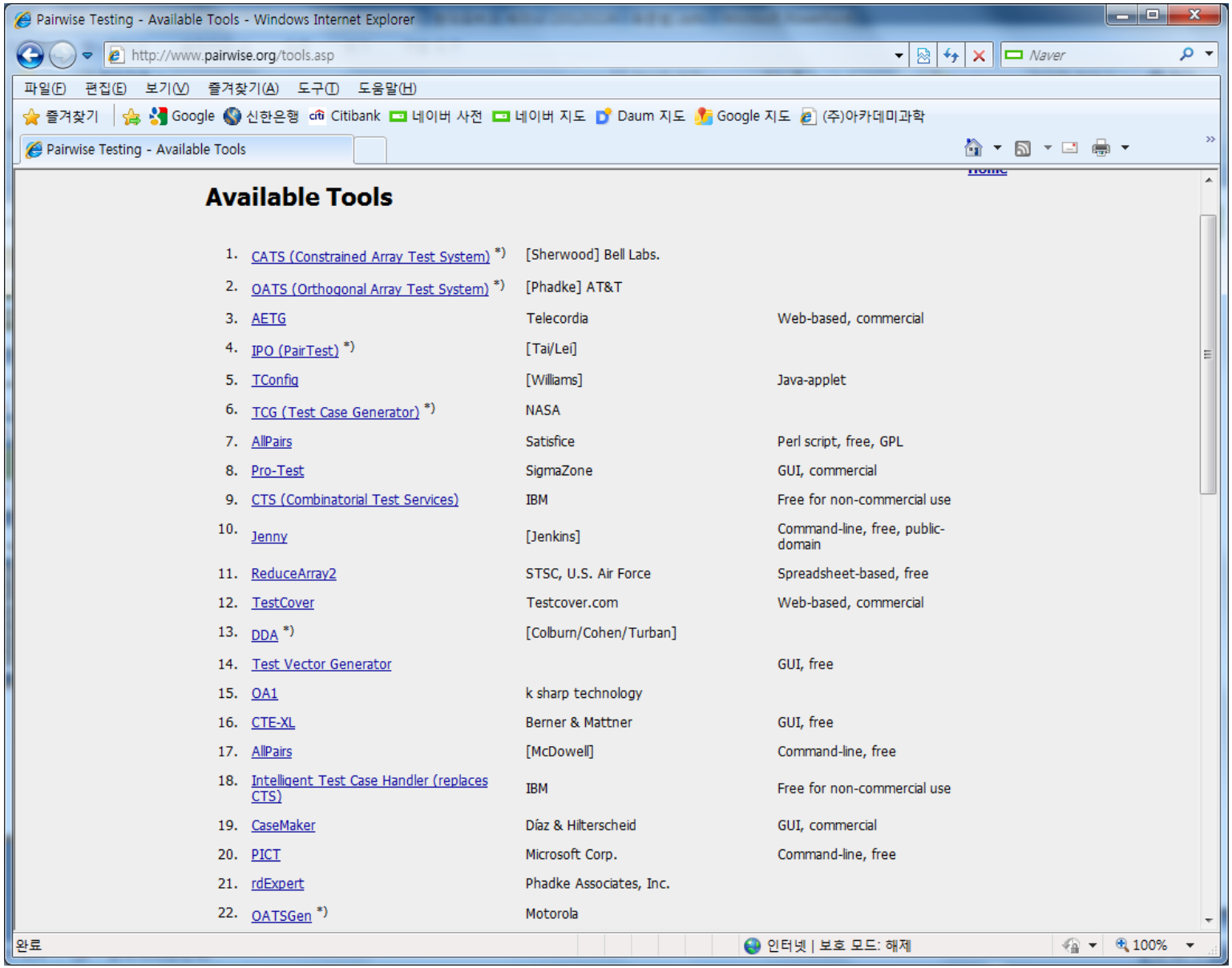
Step 3. 테스트케이스를 생성한다.

3 x 4 x 3 x 4 x 4 = 432 test cases

Pairwise Test

Test Cases: 432 → 17

Language	Color	Display Mode	Fonts	Screen Size
English	Monochrome	Full-graphics	Minimal	Hand-held
English	Color-map	Text-only	Standard	Full-size
English	16-bit	Limited-bandwidth	-	Full-size
English	True-color	Text-only	Document-Embedded	Laptop
French	Monochrome	Limited-bandwidth	Standard	Laptop
French	Color-map	Full-graphics	Document-Embedded	Full-size
French	16-bit	Text-only	Minimal	-
French	True-color	-	-	Hand-held
Spanish	Monochrome	-	Document-Embedded	Full-size
Spanish	Color-map	Limited-bandwidth	Minimal	Hand-held
Spanish	16-bit	Full-graphics	Standard	Laptop
Spanish	True-color	Text-only	-	Hand-held
Korean	-	-	Monochrome	Hand-held
Korean	Color-map	-	Minimal	Laptop
Korean	16-bit	Limited-bandwidth	Document-Embedded	Hand-held
Korean	True-color	Full-graphics	Minimal	Full-size
Korean	True-color	Limited-bandwidth	Standard	Hand-held


Pairwise Testing - Available Tools - Windows Internet Explorer
http://www.pairwise.org/tools.asp
Pairwise Testing - Available Tools
Available Tools
1. CATS (Constrained Array Test System) *) [Sherwood] Bell Labs.
2. OATS (Orthogonal Array Test System) *) [Phadke] AT&T
3. AETG Telecordia Web-based, commercial
4. IPO (PairTest) *) [Tai/Lei]
5. TConfig [Williams] Java-applet
6. TCG (Test Case Generator) *) NASA
7. AllPairs Satisfice Perl script, free, GPL
8. Pro-Test SigmaZone GUI, commercial
9. CTS (Combinatorial Test Services) IBM Free for non-commercial use
10. Jenny [Jenkins] Command-line, free, public-domain
11. ReduceArray2 STSC, U.S. Air Force Spreadsheet-based, free
12. TestCover Testcover.com Web-based, commercial
13. DDA *) [Colburn/Cohen/Turban]
14. Test Vector Generator GUI, free
15. OA1 k sharp technology
16. CTE-XL Berner & Mattner GUI, free
17. AllPairs [McDowell] Command-line, free
18. Intelligent Test Case Handler (replaces CTS) IBM Free for non-commercial use
19. CaseMaker Díaz & Hilterscheid GUI, commercial
20. PICT Microsoft Corp. Command-line, free
21. rdExpert Phadke Associates, Inc.
22. OATSGen *) Motorola

System test

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic. [1]

Activity 2063. System Testing

- Set the test data of test cases for testing
  - Example of test case 1-1 & 1-2

1 – 1	로그인 시험	존재하는 계정의 Id, pw를 입력하고 로그인 시도를 하여 로그인 기능 test	1. Login	R 1.1
1 - 2	로그인 시험	존재하지 않는 계정의 Id, pw를 입력하고 로그인 시도를 하여 로그인 기능 test	1. Login	R 1.1

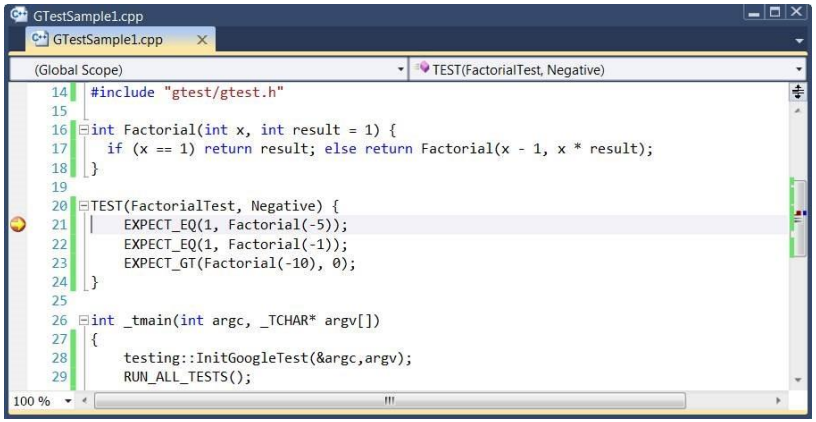
  - 1-1 : id 입력란에 test, pw 입력란에 test 입력 후 login 버튼을 누른다.
  - 1-2 : id 입력란에 test2, pw 입력란에 asdf 입력 후 login 버튼을 누른다.

Unit test

In computer programming, **unit testing** is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use.[1] Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method.[2] Unit tests are short code fragments[3] created by programmers or occasionally by white box testers during the development process. It forms the basis for component testing.[4]

유닛 테스트(unit test)는 컴퓨터 프로그래밍에서 소스 코드의 특정 모듈이 의도된 대로 정확히 작동하는지 검증하는 절차다. 즉, 모든 함수와 메소드에 대한 테스트 케이스(Test case)를 작성하는 절차를 말한다. 이를 통해서 언제라도 코드 변경으로 인해 문제가 발생할 경우, 단시간 내에 이를 파악하고 바로 잡을 수 있도록 해준다. 이상적으로, 각 테스트 케이스는 서로 분리되어야 한다. 이를 위해 가짜 객체(Mock object)를 생성하는 것도 좋은 방법이다. 유닛 테스트는 (일반적인 테스트와 달리) 개발자(developer) 뿐만 아니라 보다 더 심도있는 테스트를 위해 테스터(tester)에 의해 수행되기도 한다.

- Function/module 단위로 unit test code를 작성 후 unit testing 진행



과제는

Unit test

Unit test 개념 조사
- unit test가 무엇인지
- 수행 방법 등

Individual Practice & Assignment #3 (C Unit Test)
Team Presentation #3 (Unit Test)

C language의 unit test 및 C unit test를 위한 도구 조사
- C language를 대상으로 사용 가능한 unit test도구는 여러 가지가 있습니다.
- 팀 별로 O 개 이상의 도구를 조사 후, 그 중 하나를 골라 test code 작성법 등 상세히 작성하시면 되겠습니다.
- 추후 발표 때 각 팀에서 선정한 도구를 이용해서 unit test 를 진행 후 발표를 진행하도록 하겠습니다.

팀 별로 word를 이용해서 보고서 작성하여 10/31 까지 pdf 로 변환하여 제출
(양식은 자유롭게 제출해 주세요)