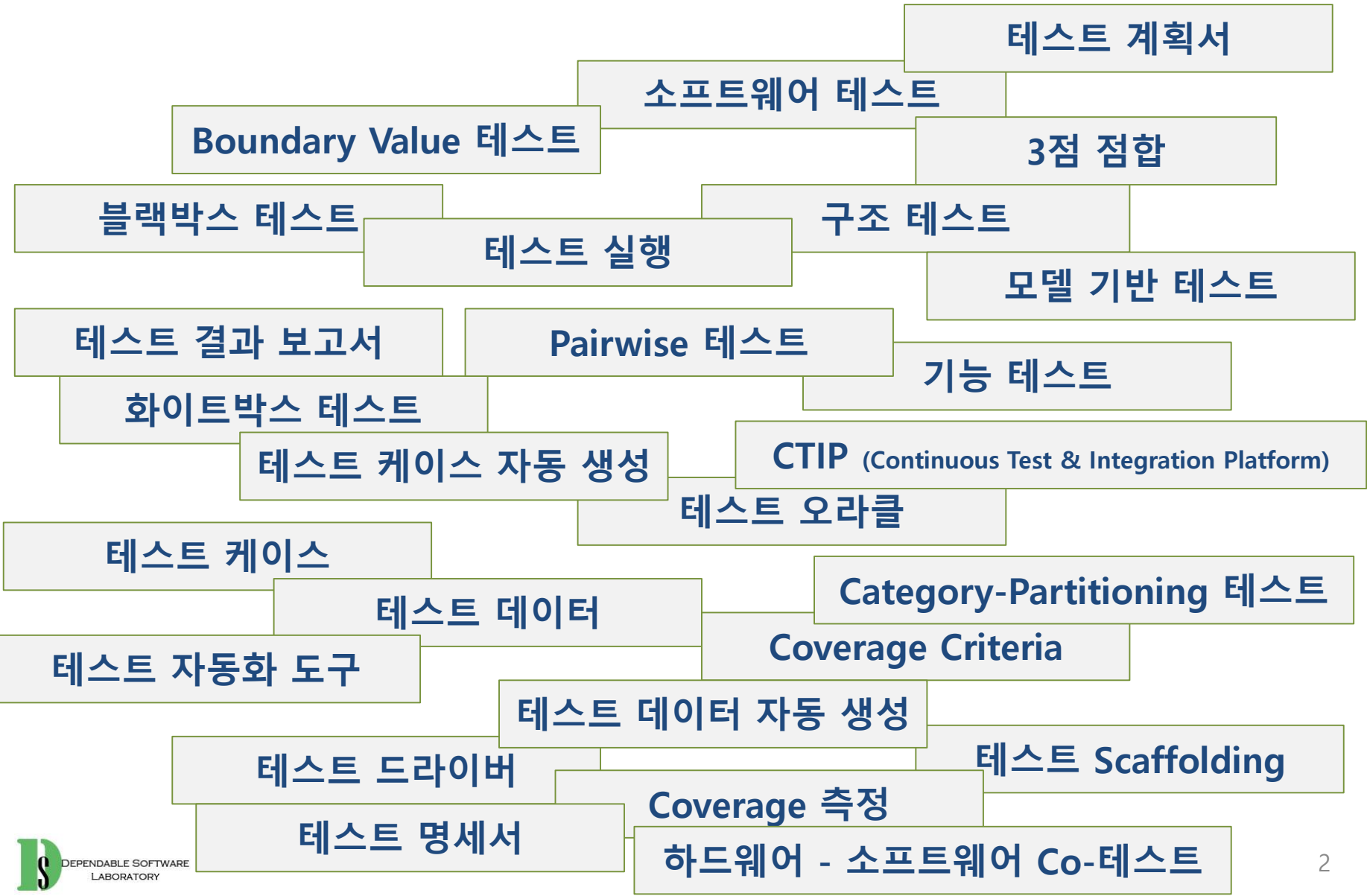


# 소프트웨어 테스트

- 개념, 기법 및 활용 -

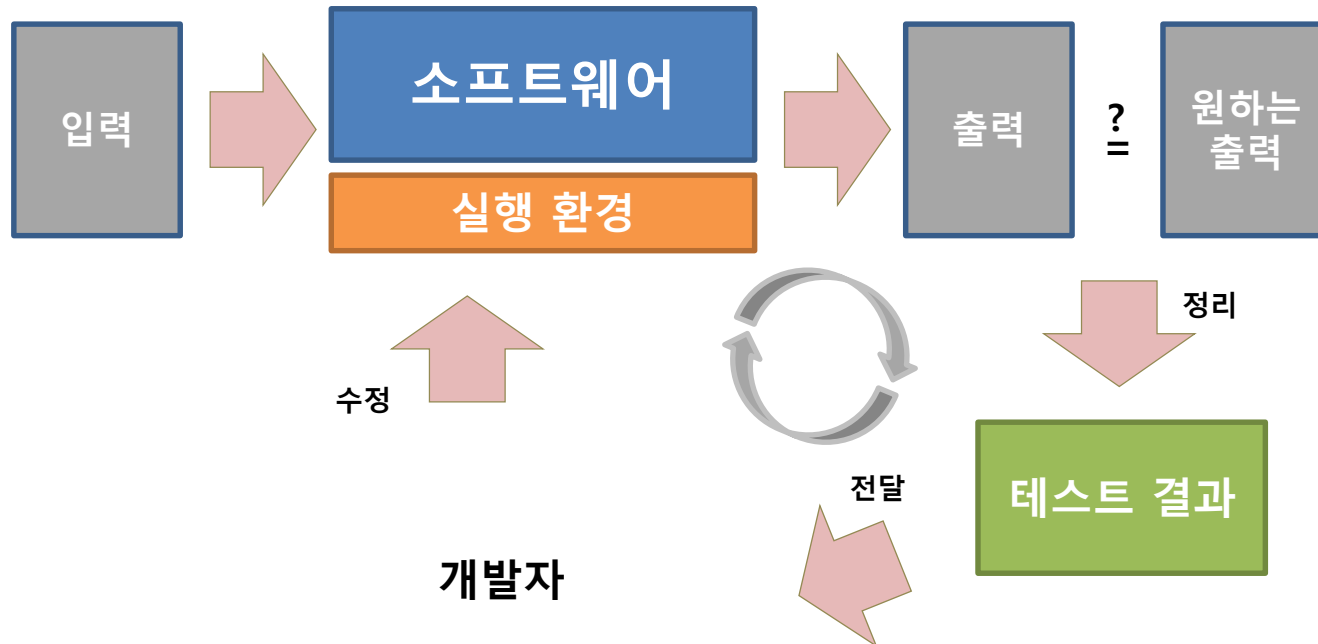
Dependable Software Laboratory  
손준익

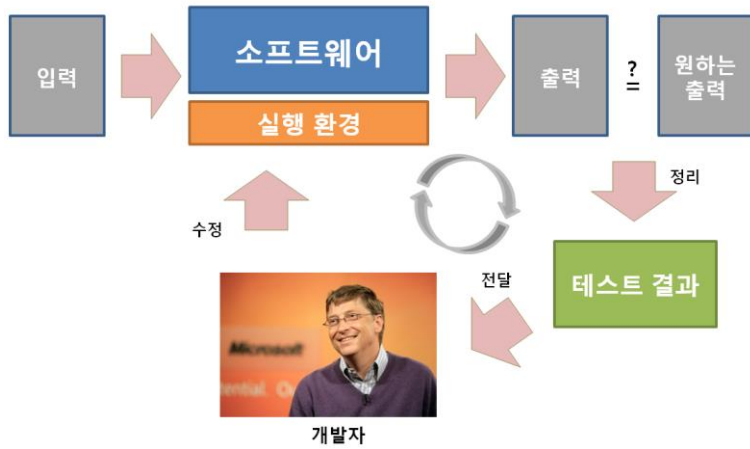
# 테스트 분야 개념들



# 소프트웨어 테스트

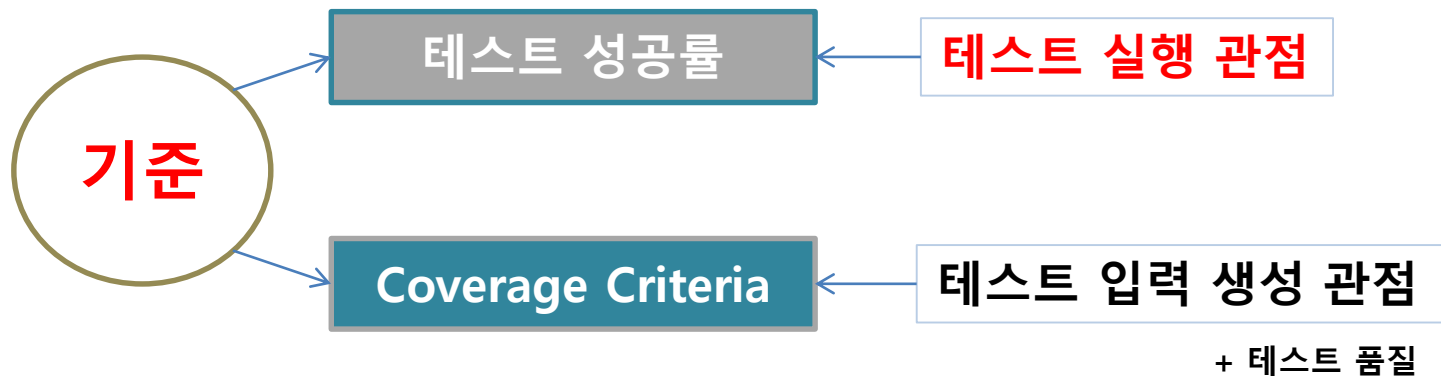
입력을 주고 소프트웨어를 직접 실행함으로써, 원하는 출력이 생성되는지 확인하는 일련의 작업





# 언제까지 반복?

정답: **기준**을 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

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

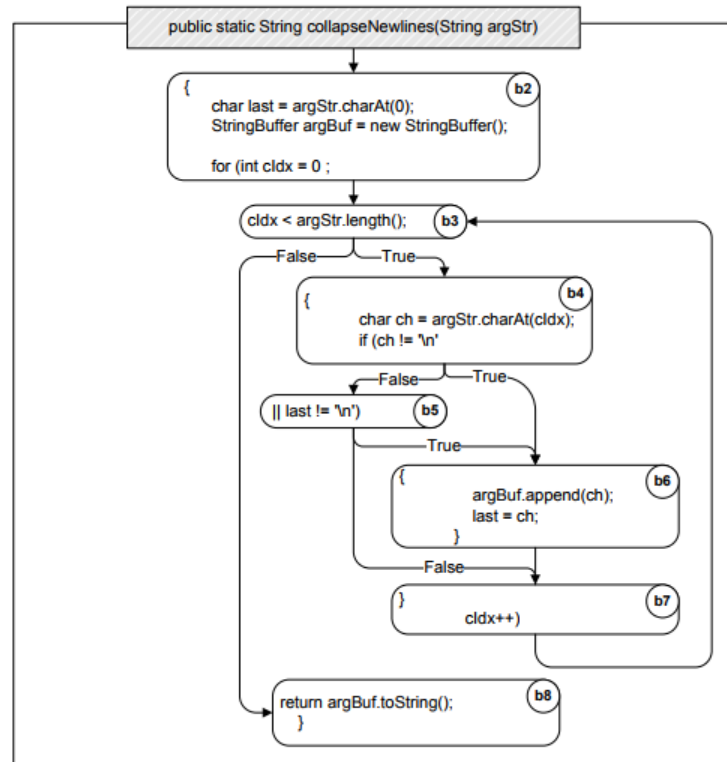
# 화이트박스 테스트

## CFG 예제

```
public static String collapseNewlines(String argStr)
{
    char last = argStr.charAt(0);
    StringBuffer argBuf = new StringBuffer();

    for (int cldx = 0 ; cldx < argStr.length(); cldx++)
    {
        char ch = argStr.charAt(cldx);
        if (ch != '\n' || last != '\n')
        {
            argBuf.append(ch);
            last = ch;
        }
    }

    return argBuf.toString();
}
```



출처 : <http://dslab.konkuk.ac.kr/Class/2015/15SV/Lecture%20Note/Software%20Testing%20and%20Analysis.pdf>

# 질문!!!

1. 시스템시험(System Test)은 어떤 종류의 테스트입니까?

정답: 블랙박스 테스트

2. 단위시험(Unit Test)은 어떤 종류의 테스트입니까?

정답: 두 종류 모두 가능

# 블랙박스 테스트





**Haystack**



**Needles**



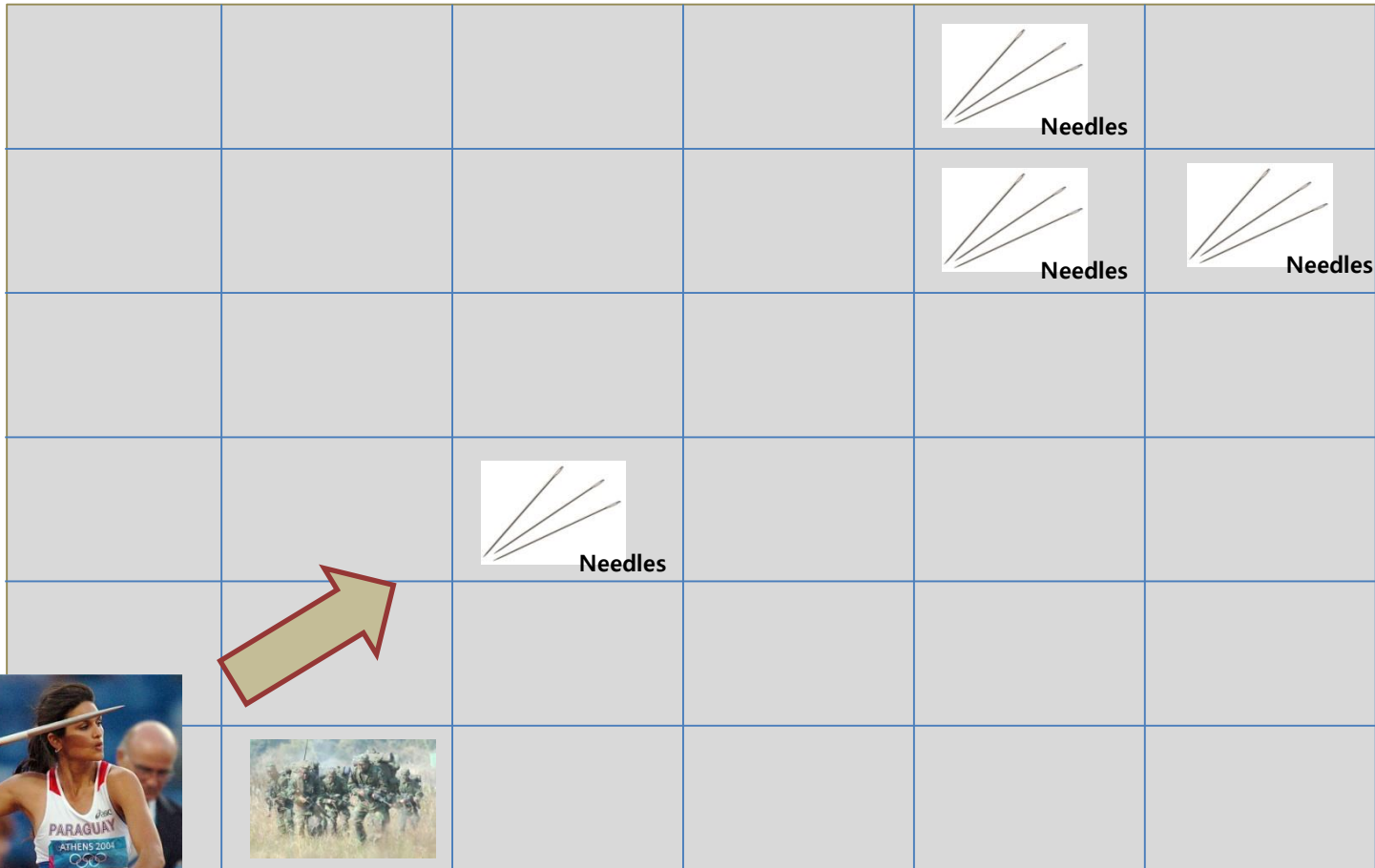
**Let's find needles out !!!**



# 균등 분할

= Random Testing  
 = Uniform Testing


# 실제로 바늘은?



# 바늘을 신속하게 찾기 위해서는?

정보 (Information)



누가?

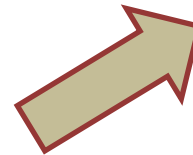
개발자 누가?



Needles

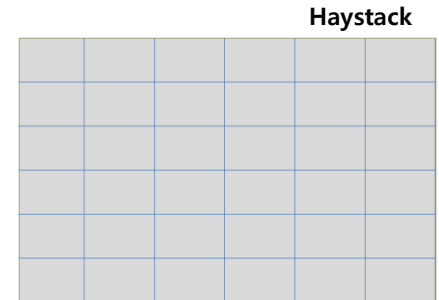
무엇을?

어떤 기능을?



어디로?

어떤 방식으로?



Haystack

어디에서?

어떤 환경/가정에서?

# 정보는 어디에?

## 명세서 (Specification)

프로젝트 계획서 (Project Plan)

요구사항 명세서 (Requirements Specification)

디자인 명세서 (Design Specification)

테스트 계획서 (Test Plan)

각종 모든 문서

# 블랙박스 테스트

명세서에 정의된 **기능**이 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)

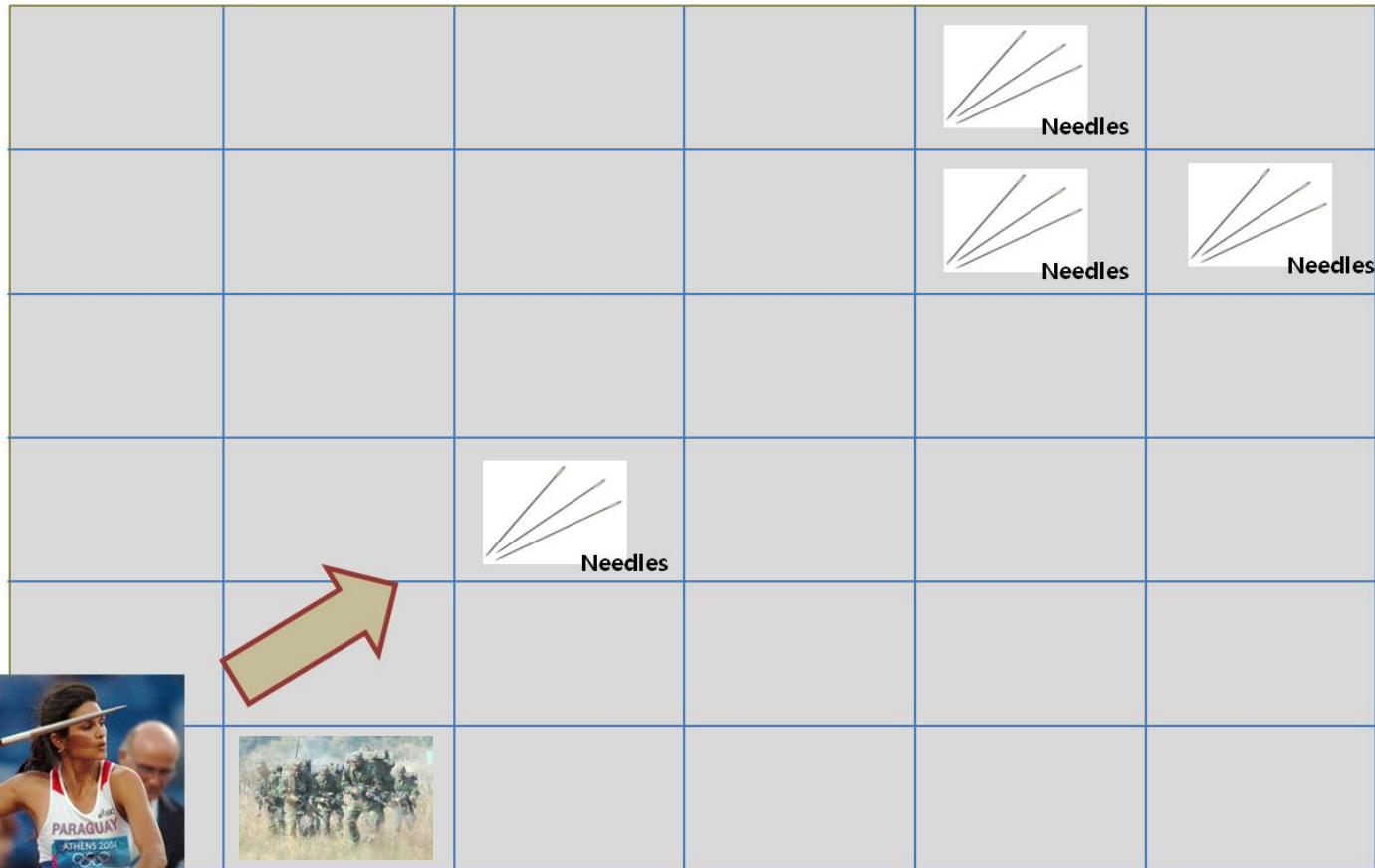
특정 정보를 이용하여, 전체 테스트 입력 범위 중 오류를 잘 찾아낼 수 있는 입력구간(region of input space)을 찾아낸다.

∴ 전체 입력 범위를 모두 테스트 할 수 없다.

∴ 오류들은 주로 몰려있다.

# 균등 분할


동일한 Cost로 수행되는 테스트 범위

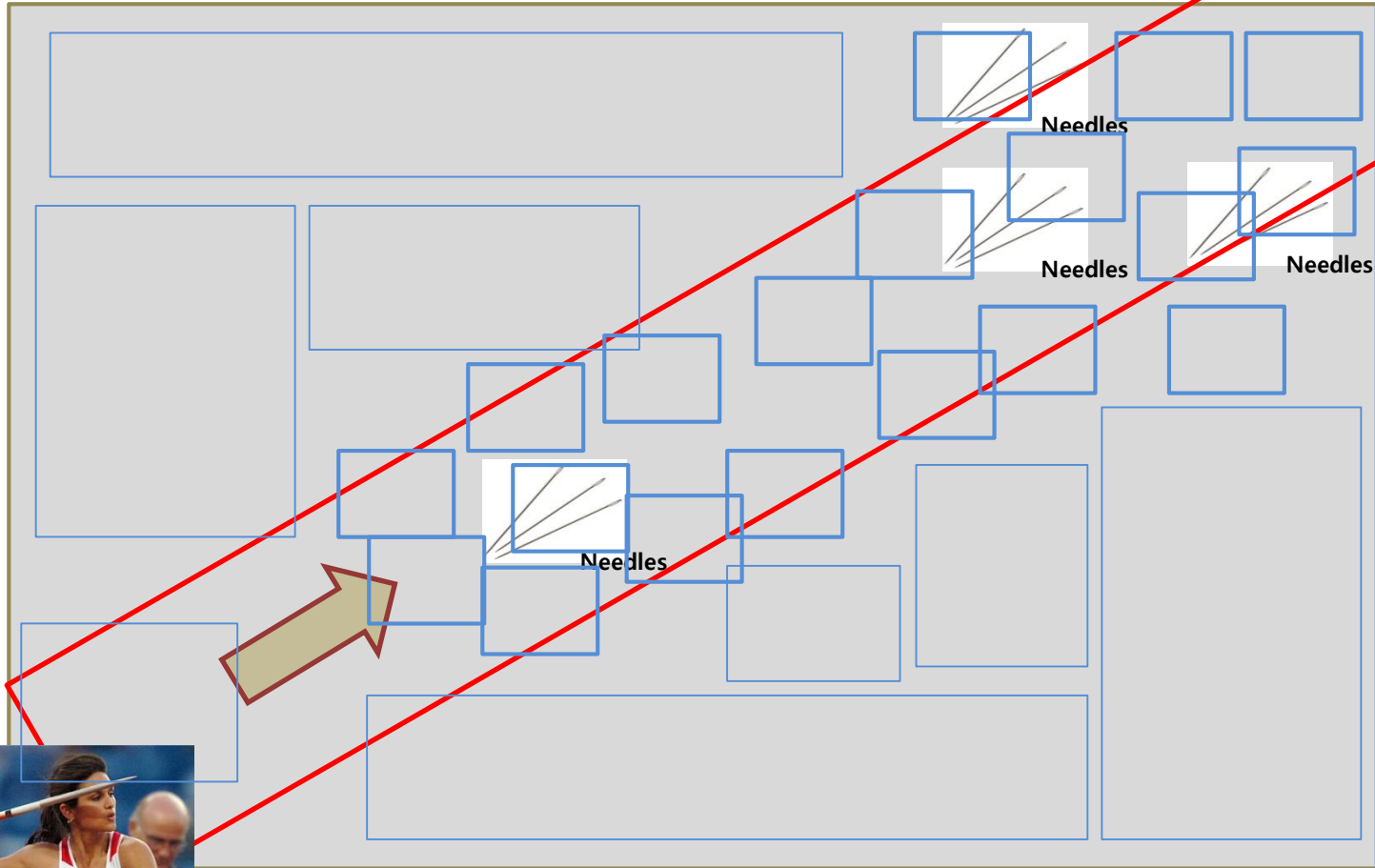


테스트 비용(회수): 36회  
테스트 효율: 低



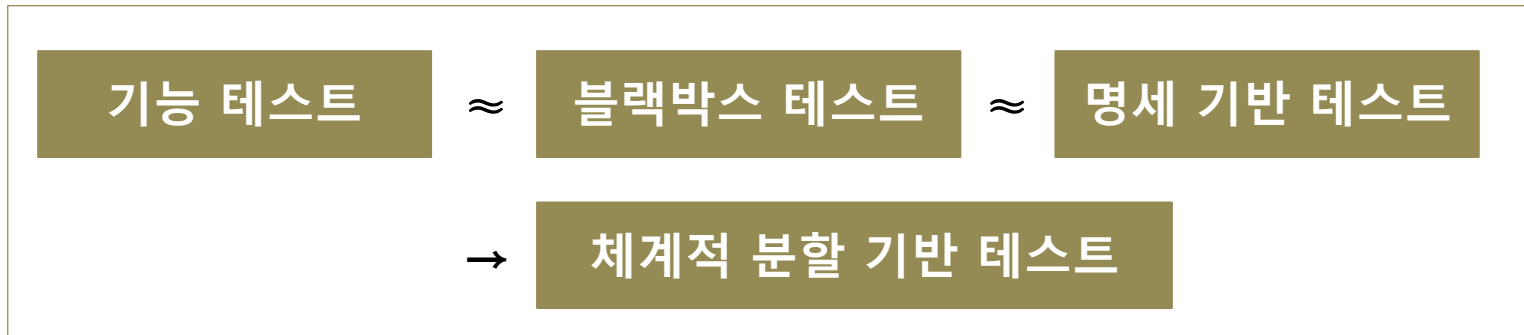
# 체계적 분할

 동일한 Cost로 수행되는 테스트 범위



테스트 비용(회수): 27회  
테스트 효율: 高

# 기능 테스트

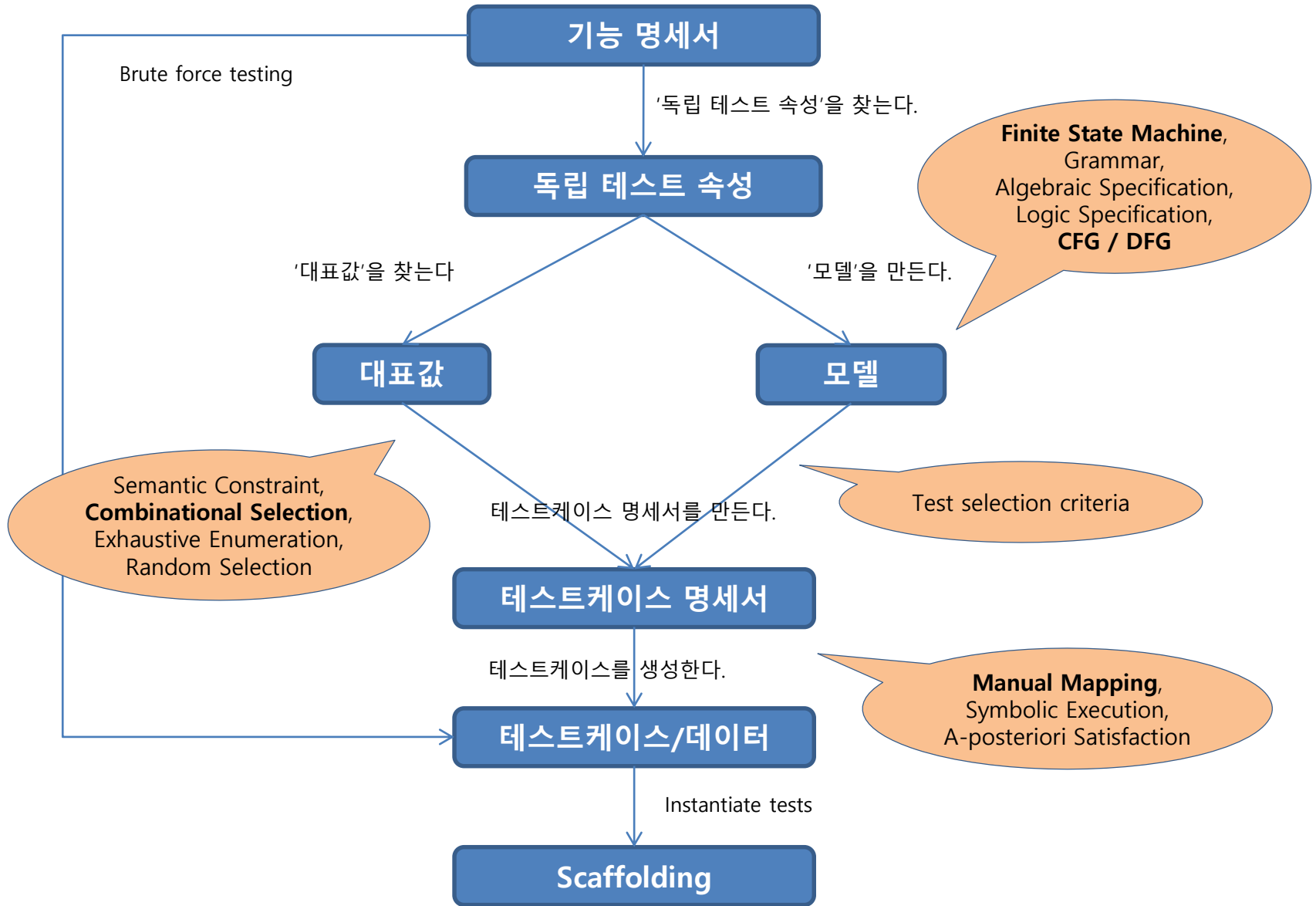


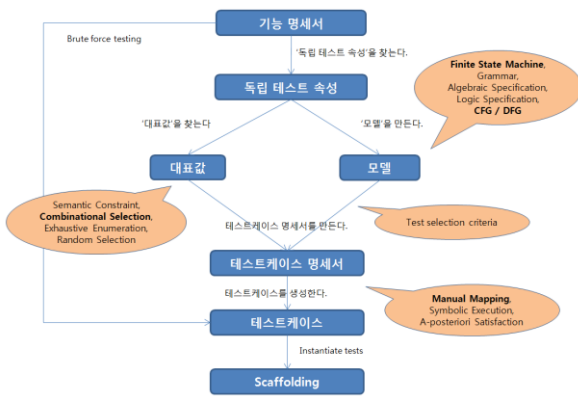
명세(Specification)를 사용하여 입력 범위를 분할(partitioning of input space)



각 분할된 범위(Category)를 테스트

# 기능 테스트 단계





## 기능명세서 분석

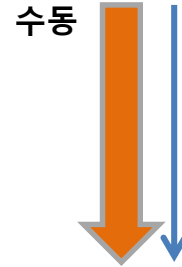
## 전통적인 블랙박스 테스트

## 모델 기반 테스트

## 테스트 데이터 생성

어려워요!!!

기능명세서 분석



수동

수동

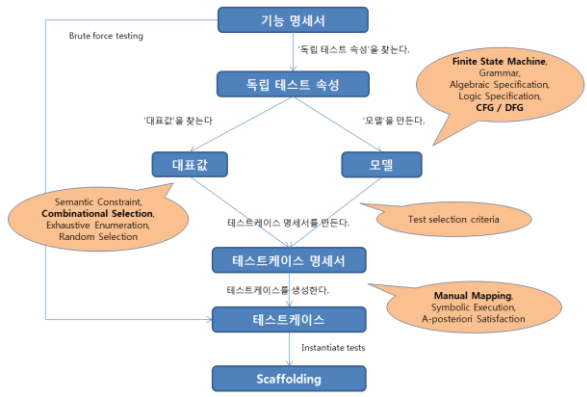
수동

자동

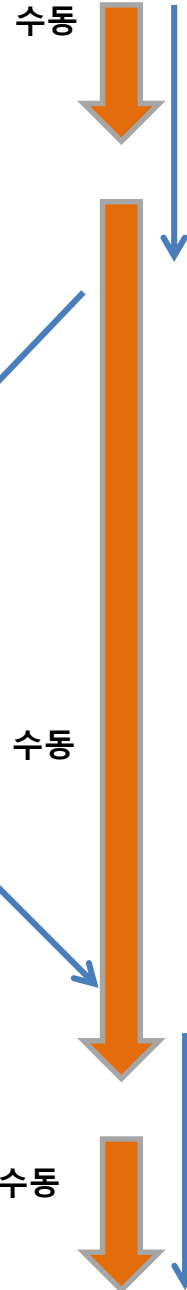
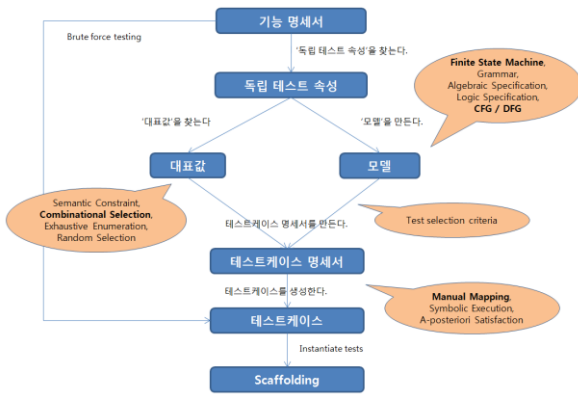
테스트 데이터 생성

모델 기반 테스트

전통적인  
블랙박스 테스트



# 현실은?



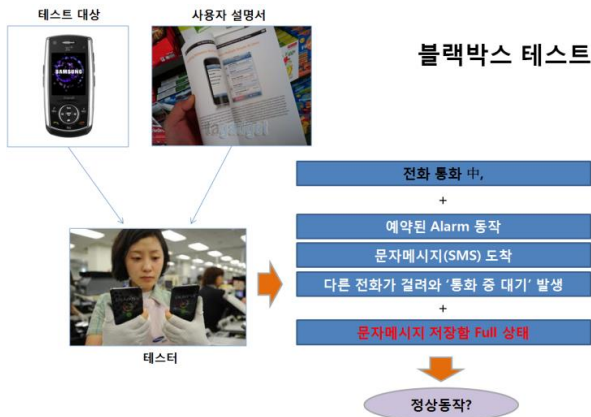
기능명세서 분석

전통적인  
블랙박스 테스트

모델 기반 테스트

Brute Force Test  
→ 막 테스트

테스트 데이터 생성



# 기능 테스트를 잘 하기 위해서는?

1. 기능명세서를 켜자.

삼성전자 T직군 SW 테스터처럼

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.



# Category-Partitioning Test

예제: 모니터 디스플레이 테스트

Step 1.

'독립 테스트 속성(Category)'을 찾는다.

Display	Language	Fonts	Color	Size
---------	----------	-------	-------	------

Step 2.

'대표값 (representative value)'을 찾는다.

Display Mode	Language	Fonts	Color	Screen Size
Full-Graphics	English	Arial	Monochrome	Small
Text-Only	French	Roman	256 gray	Laptop
Limited-Bandwidth	Spanish	Calibri	16-Bits	Full-Size
	Korean		32-Bits	

Step 3.

테스트케이스를 생성한다.

$$3 \times 4 \times 3 \times 4 \times 4 = 432 \text{ test cases}$$

Test Cases: 432 → 17

# Pairwise Test

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

완료

인터넷 | 보호 모드: 해제

100%

# 과제

## Unit Test

### Unit test 개념 조사

- unit test가 무엇인지
- 수행 방법 등

### C language의 unit test 및 C unit test를 위한 도구 조사

- C language를 대상으로 사용 가능한 unit test 도구는 여러 가지가 존재
- 개인 별로 1개 이상의 도구를 조사 후,  
그 중 하나를 골라 test code 작성법 등 도구와 관련된 내용을 상세히 작성
- unit test 발표 때, 각 팀에서는 조사한 도구 중 하나의 도구를 선정하여 unit test 진행 후 발표

Word 또는 한글을 이용하여 보고서 작성하여 11/6 자정까지 pdf 형식으로 제출

- 메일 제목: [2017SE\_B][Assignment#3]학번\_이름
- 파일 제목: [2017SE\_B][Assignment#3]학번\_이름.pdf