CSE4006 Software Engineering

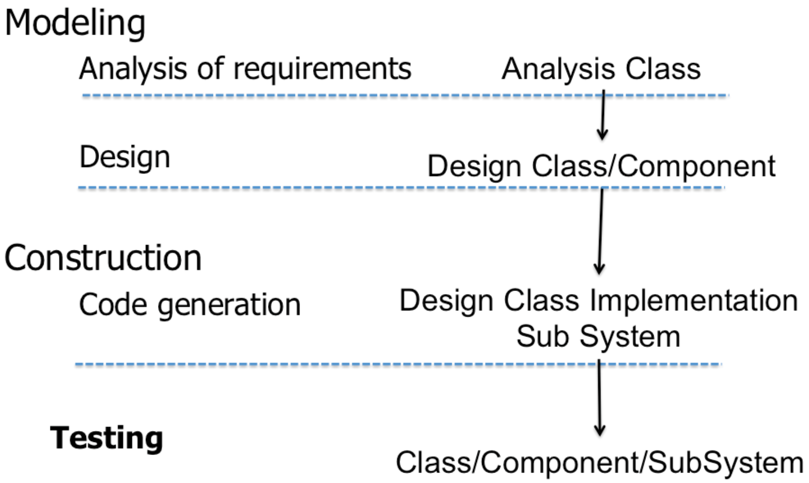
10. Software Testing Fundamental Concepts

Scott Uk-Jin Lee

Department of Computer Science and Engineering
Hanyang University ERICA Campus

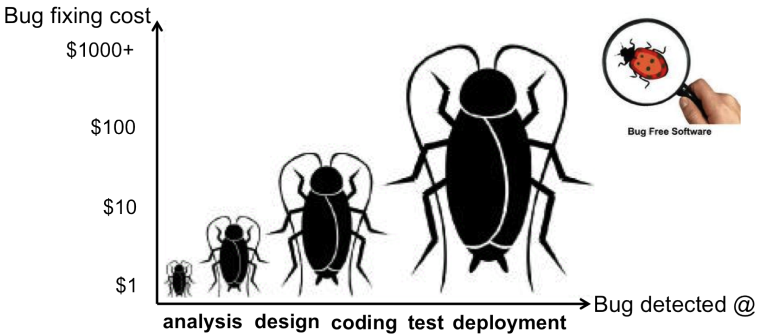
1st Semester 2015

Testing in Object-Oriented Point of View



Error Correction Cost

- Software Quality ≈ Defect prevention ⇒ apply SE Techniques
- Testing is essential to remove introduced defects



lab(se);

Software Quality & Testing

- Software testing is a task of detecting defects through the execution on computer
- Software became an important element in real-time embedded systems and various other areas
  $\Rightarrow$ demand for software quality has increased
  - in order to maintain the desired level of software quality, defects should not be introduced into software during the development process
  - software testing is required as a tool to remove defects introduced in software

Software Quality & Testing

- Testing is a task of checking whether the software is developed as intended
  - test design: task of finding the most ideal input value for testing
  - testing input is not to obtain an output but to detect defects
- For accurate testing requirements specification must exist
  - all user requirements should be accurately reflected
  - must be detailed enough to be accurately reflected in the code
- Testing is one of the method for **Quality Assurance**

Software Quality Factors

- Because it is difficult to clearly define the quality of software, various quality indicating factors are considered
- ISO/IEC 9126 Quality factors
  - replaced with ISO/IEC250mn (SQuaRE)
    - Functionality
    - Reliability
    - Usability
    - Efficiency
    - Maintainability
    - Portability
- McCall's Quality Factors (1977)

McCall's Quality Factors

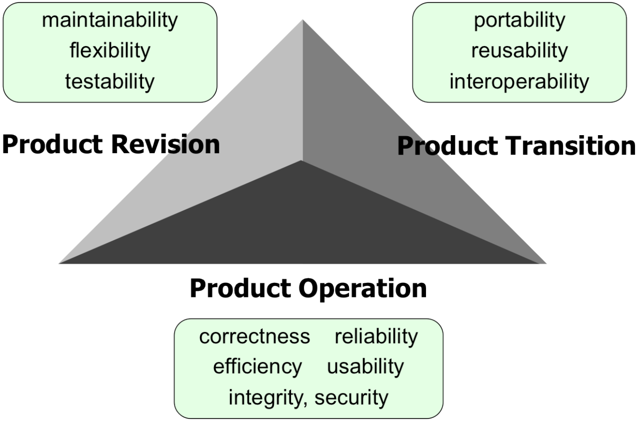
- Defines two main concepts of software quality
- Quality factor
  - represents behavioral characteristics of system

    e.g. correctness, reliability, efficiency, testability, poratbility, ...
- Quality criteria
  - properties of quality factor associated with software development

    e.g. modularity is a property of software architecture

    - well-modularized software groups coherent components into a module to increase maintainability of system

McCall's Quality Factors



McCall's Quality Factors

- **Product Operations** : Quality factor indicating operational suitability
  - **Correctness** : extent to which an implemented software satisfies its specifications
  - **Reliability** : extent to which an implemented software works without failure
  - **Efficiency** : how efficiently an implemented software performs its functions
  - **Integrity** : extent to which access by unauthorized person can be controlled
  - **Usability** : effort required to operate software

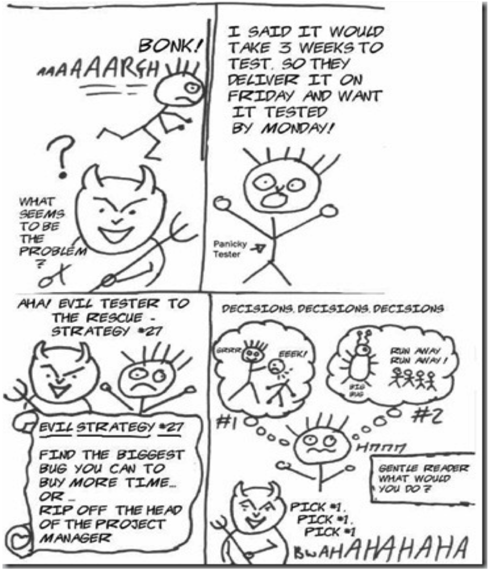
McCall's Quality Factors

- **Product Revision** : factor indicating ease of modification
  - **Maintainability** : effort required to fix a defect
  - **Flexibility** : effort required to modify an operational software
  - **Testability** : effort required to test whether intended function is performed
- **Product Transition** : quality factor indicating ease of increasing utilization
  - **Portability** : effort required to transfer a software to another software or hardware environment
  - **Reusability** : extent to which part of software can be reused in other application
  - **Interoperability** : effort required to couple one software to another

Reasons for Difficulties in Testing

- Software Complexity
- Incomplete Specification (Requirements Spec. / User Guide)
- Difficulties in establishing the operational environment for testing
- Problems due to the unique characteristics of software
  - very minor mistake/error that are very unlikely
    $\Rightarrow$ serious consequences (e.g. Therac-25, Ariane 5)
- Absence of test mind
- **Software defect prevention = apply SE techniques**
  - reduces mistakes in the development & assist in testing
    e.g. structured programming, modular design techniques

Evil Tester explains ... Find the Big Bug First


BONK!
aaaAAAARGH
WHAT SEEMS TO BE THE PROBLEM ?
I SAID IT WOULD TAKE 3 WEEKS TO TEST. SO THEY DELIVER IT ON FRIDAY AND WANT IT TESTED BY MONDAY!
Panicky Tester
AHA! EVIL TESTER TO THE RESCUE - STRATEGY #27
EVIL STRATEGY #27
FIND THE BIGGEST BUG YOU CAN TO BUY MORE TIME... OR... RIP OFF THE HEAD OF THE PROJECT MANAGER
DECISIONS, DECISIONS, DECISIONS
GRRR
EEEK!
BIG BUG
RUN AWAY RUN AWAY!
#1
#2
HMMM
GENTLE READER WHAT WOULD YOU DO ?
PICK #1, PICK #1, PICK #1
BWAHAHAHAHAHA

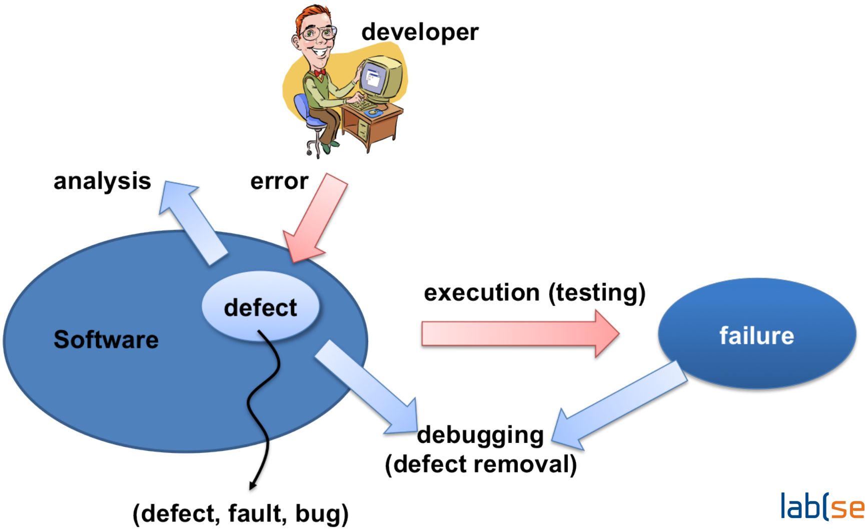
Testing Overview

- Testing is a running of a program to detect defects
  - if defects are not detected, test is failed
- Goal of testing:
  - **Verification** : verifies that a software is implemented exactly as specified in the specification by showing that all the execution results are not different from the expected results
  - **Validation** : before delivering the completed software to a customer, checks whether the software operates correctly and satisfies all the customer's required functionalities and constraints when installed and running on an actual operation environment

Testing Overview

- Testing Process
  - tester interprets part of specification that corresponds to the code and represent it in the form of **test case**
  - execute test case for the code
  - check whether the results are correct through test oracle
- Test Oracle
  - mechanism used to determine expected result with the reliable value(input) extracted from the specification
  - essential to determine the accuracy of execution results
  - **testing is checking whether oracle and value obtained from the test execution matches**

Software Defects



lab(se);

Software Defects

- Defect or Fault
    - all actions of software product that does not match defined characteristics
- Detected defect
    - defects detected before software is installed/operational
- Residual defect
    - defects passed onto the installed/operational environment
    - defects that are not found before the installation or found but did not removed
- Software failure
    - a set of abnormal symptoms occurring during the operation due to the potential software defects

lab(se);

Testing Types - Classification by Purpose

- The ultimate goal of testing is to check whether customer's requirements are satisfied or not
  - implemented program satisfies protocol, standard, requirements contract ...
- Defect test
  - test conducted for the purpose of fault detection
- Validation test / Conformance test
  - performance test, usability test, safety test, etc

lab(se);

Testing Types - Test based Classification

- according to what basis are used when designing a test
- **Specification-based Test** = **Black-box Test**
  - examine relationship between input used for the execution and output produced from the execution without considering the code contents
  - also known as **functional test**
- **Code-based Test** = **White-box Test**
  - analyze the structure and the logics of code
  - also known as **structural test**
- Scenario-based Test = Purpose-based Test
  - mainly examine the functions using usage scenarios

lab(se);

Testing Types - Classification by Test Design Techniques

- Systematic Test
  - to devise test cases that best detects defects
  - **Sentence test** : examine every sentence that exists in the program at least once
  - **Branch test** : examine every program branch at least once
- Random Test
  - uses test cases that are randomly generated (without a specific test data selection method)
  - can calculate test success rate $\rightarrow$ can apply statistical meaning to reliability

Testing Types - Classification by Test Level

- at what point in the development process, the test is executed, according to the lifecycle model

- Module test = Unit test
- Integration test
- System test
- Acceptance test
- Installation test = Field test
- Regression test

Test Execution Example

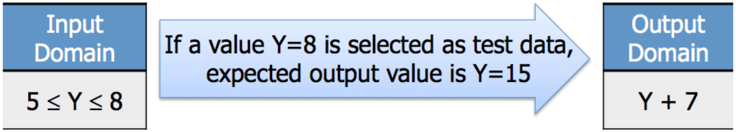
- depending on the value of the input Y (a single integer), outputs specified value added to the input integer value
- input domain are divided into four parts

Input Domain	Output Domain
$Y > 8$	$Y + 5$
$5 \le Y \le 8$	$Y + 7$
$1 < Y < 8$	$Y + 5$
$Y \le 1$	$Y + 3$

```
if(Y > 1)
    Y = Y + 1;
    if(Y > 9)
         Y = Y + 1;
    else
          Y = Y + 3;
    Y = Y + 2;
else
    Y = Y + 4;
if(Y > 10)
    Y = Y + 1;
else
    Y = Y - 1;
```

Black-box Text Execution Process

- input test data prepared according to the specification
    - examine if the execution result matches the expected result
- execute test for each part of input domain



If a value Y=8 is selected as test data, expected output value is Y=15

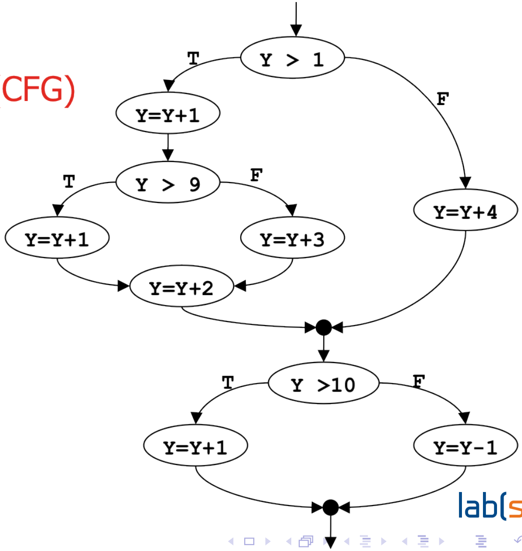
Execute code with Y=8 as an input

```
if(Y > 1) Y = Y + 1;        (Y = 9 )
    if(Y > 9)
    else Y = Y + 3;         (Y = 12)
    Y = Y + 2;              (Y = 14)
if(Y > 10) Y = Y + 1;       (Y = 15)
```

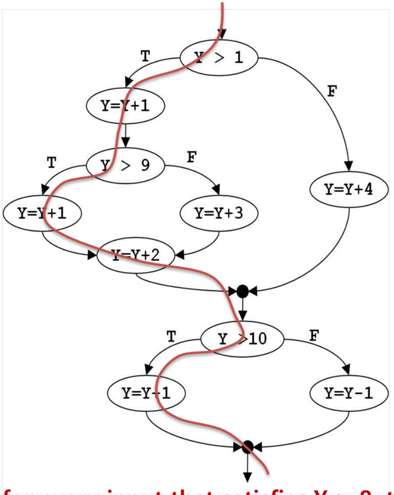
White-box Text Execution Process

Control Flow Graph (CFG)

```
if(Y > 1)
    Y = Y + 1;
    if(Y > 9)
         Y = Y + 1;
    else
          Y = Y + 3;
    Y = Y + 2;
else
    Y = Y + 4;
if(Y > 10)
    Y = Y + 1;
else
    Y = Y - 1;
```



White-box Text Execution Process



Program Path			Path Space
T	T	T	**Y > 8**
T	T	F	cannot execute
T	F	T	$5 \leq Y \leq 8$
T	F	F	$1 < Y < 5$
F	-	T	cannot execute
F	-	F	$Y \leq 1$

for every input that satisfies Y > 8, this path is executed

Limitations of Testing

- depending on the size of the code, the number of path increases dramatically
  - for repeating structure, countless number of path may exists
  - examining with all possible input value is virtually impossible
  - realistically limiting the input scope → huge drop on defect detection probability
- The goal of software testing is NOT determining the correctness, but increasing the efficiency of defect detection
  - Dijkstra : Testing is an efficient means for showing the presence of error in the code but, hopelessly insufficient to prove the absence of error
  - Beizer : Pesticide Paradox

Software Test Process

- Test Process
  - **Black-box** test : determine input value to be used in the test and examine the output value resulted from the execution
  - **White-box** test : prepare appropriate input values for many execution path existing within the program and execute
- There are many input value for path execution
  - **Path domain** : a set of input value that executes the same path
  - **Path computation** : output value obtained from execution of the determined path
  - Test data : input values specifically selected from the path space for execution

Software Test Process

Test execution Path

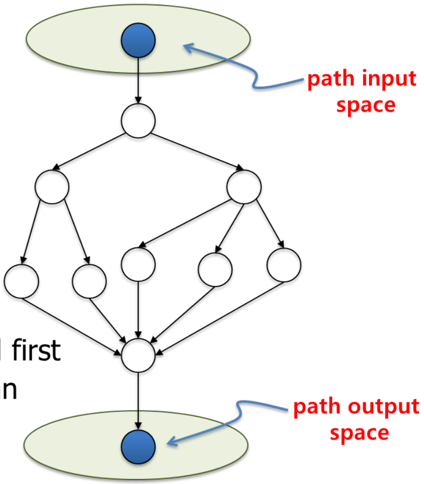
- Test execution path is decided by input value

Black-box Test

- select input value first
- execute program
- examine output value

White-box Test

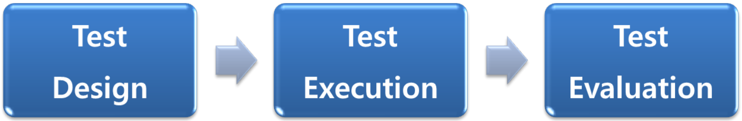
- select path to be examined first
- prepare input value that can execute the selected path



lab(se);

Software Test Proess

- data values of a single path space have the same or slightly different defect detection effectiveness
- Equivalent class
  - data space with the same defect detection effectiveness
  - a single test for a path is enough since the defect detection effectiveness is the same
  - very efficient as the entire space is tested at once
- Test Process



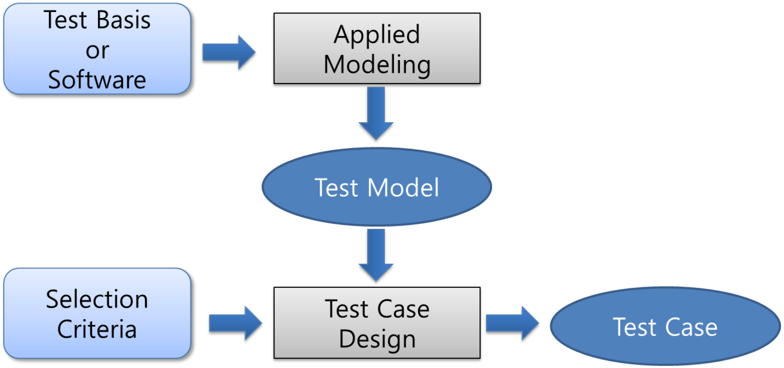
Test Design

- Stage where test data is prepared for detecting defects
- **Test Case** : specific input values to be used in the test
    - extract based on specification or code, according to test data selection criteria
- **Test Basis** : resource referenced for extracting a test case, such as specifications or code
- Test Model : apply modeling techniques since extracting test cases directly from the test basis are difficult
    - control flow graph of a program is a typical test model

lab(se);

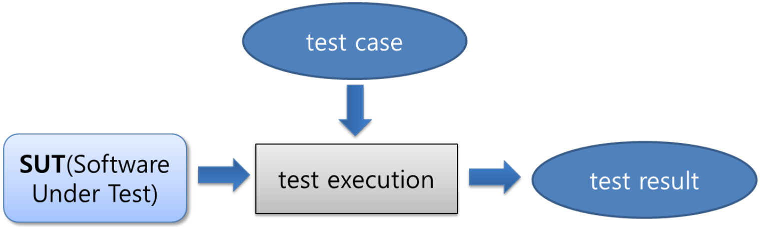
Test Design

- Test case design procedure

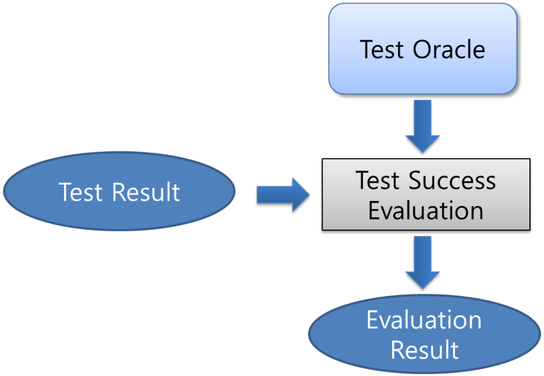


Test Execution

- Test driver : automated tool used for efficient test execution

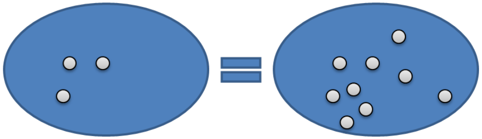


Test Result Evaluation



Test Case Design

- Test case design = Test design
- The entire test process level is determined by the test cases
  - each test case must have high probability of detecting errors
  - must use as small number of test cases as possible



Test effort according to the number of defects

Test Case Design

- Test Case Design Process
  - collect test clue on defects of SUT
  - define the clue as specific test requirements
  - for each test item, write test specification
    - what a test case trying to investigate
    - input condition required for test case execution
    - expected output of test case execution
- Test scenario : definition of execution order of several test cases
- **Test Script** : detailed description of test execution procedures in formal test specification language

lab(se);

Test Oracle

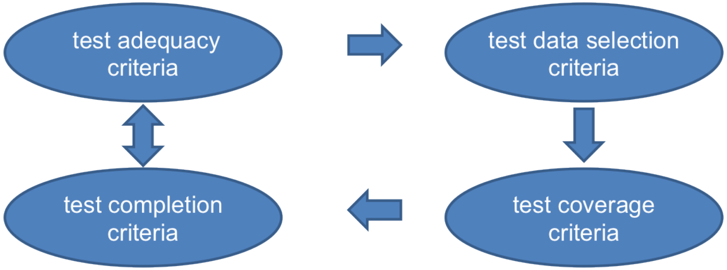
- Finding out and obtaining expected results in advance is essential to test design
- Complete oracle should show exactly the same behavior as the correctly implemented system

Test Criteria

- The principle of test is to minimize test cases while maintaining sufficient level of coverage
- Test Criteria : extent to which a test is determined to be sufficient
  - depends on the required level of quality
  - derived from the customer's requirement specifications or code
- **Test Predicate** : test criteria described in the form of predicates
  - includes condition who's result is either 'True' or 'False'
  - in the form of single or complex condition

Test Criteria

- Types of test Criteria
  - Adequacy criteria
  - Data selection criteria
  - Coverage criteria
  - Completion criteria


test adequacy criteria
test data selection criteria
test coverage criteria
test completion criteria