

Vietnam and Japan Joint ICT HRD Program

ITSS Software Development **Chapter 12-13. Software Testing**

Nguyen Thi Thu Trang
trangntt-fit@mail.hut.edu.vn

1

Content

- ⇒ 1. Testing Strategy
- 2. Testing Techniques
- 3. Statistical testing method
- 4. Testing Method for Case Study
- 5. Other viewpoint for Test
- 6. Test Planning
- 7. Whole Testing Procedure in a Project

2

1.1 What software defect can be detected in what type of testing

In V-formed figure, show the relationship between defect building stage and defect detection stage. Only corresponding detection stage test can detect the defects.

| <Building Stage of SLCP> | | <Detection Stage> |
|--------------------------|--------|-----------------------|
| Requirements Analysis | -----> | System Test |
| Architectural design | -----> | Program Test |
| Detailed Design | -----> | Unit Test |
| Construction/Coding | -> | Code Review/Unit test |

Unit Test: To test/confirm the logic of each unit is realized.

Integrated Test: To test/confirm the architecture of the software is realized.

System Test: To test/confirm the requirements of the system is realized.

Note: The Software Qualification Test can be included in System Test in System Development Process.

1.2. Basic Policy for Each Test Phase

-The defects detected in System Test, or Integrated Test, need more man- hour to correct them more than those detected in Unit Test.

→In Requirements Analysis process, Architectural design process, and Detail design process, the reviews are needed detect and correct the defects.

-Code Review/Unit Test (including DB test) determine coding quality.

→Unit Test should cover all External Specification of the Unit.

→Unit Test should cover Code coverage.

1.2. Basic Policy for Each Test Phase (2)

- Program/Software Test determines software design quality.
 - Integrated Test should cover all External Specification of the software.
 - Integrated Test should cover Unit/Module coverage.
- System Test determine System quality.
 - System Test should cover all Software Requirements including Qualification Requirements.
 - System Test should include non-functional tests.

Content

1. Testing Strategy
- ⇒ 2. Testing Techniques
3. Statistical testing method
4. Testing Method for Case Study
5. Other viewpoint for Test
6. Test Planning
7. Whole Testing Procedure in a Project

2.1 Basic Idea for efficient and sufficient test

“Cover all specification to be tested!!”→Of course?

Each test should be achieved covering corresponding all specification.
For example, Unit Test is achieved covering “The Unit specification.”

How to realize? Possible? Too many Test cases are needed?



Equivalent Analysis is available for creating efficiently the test cases.

“Equivalent Analysis”: the specification is divided into equivalent area of inputs or conditions which result in the equivalent result. And take one test case in one equivalent area.

“Cover all codes to be tested!!”→Of course?

If all specification are tested, then all codes are tested?

How to realize? Possible? →All codes? All branches? All iterations? Too many Test cases are needed?



Test Coverage and the Tools are available for achieve this kind of test.

2.1 Basic Idea for efficient and sufficient test (2)

Focus on important characteristics/aspects!!

The places where defects exists should be intensively tested!! - Possible?



Boundary Analysis is typical way for this!!

Boundary Analysis: Near boundary, many defects exist especially “data input analysis facility”.

Focus on important “Non-functional Specification”!!



Non-functional specification the customer specified should be tested sufficiently by priority.

2.2 Another Idea for Efficient and Sufficient test

Focus on used the design method and notations,
For Structured design, 


- Control Flow Test / Condition Expression Test
- Data Flow Test
- State Transition Test

For Object Oriented Method, 

The test methods mentioned above are available. And additional tests methods below are needed.

- class linked test
- UML based Test
- Use Case based Test
- Overwrite test / dynamic linked test

Content

1. Testing Strategy
2. Testing Techniques
-  3. Statistical testing method
4. Testing Method for Case Study
5. Other viewpoint for Test
6. Test Planning
7. Whole Testing Procedure in a Project

3.1. Categorization of Testing Method based on Test Specification

Testing methods can be categorized according to the specification on which their test cases are created based.

External Specification Test: The test cases of this type test are created based on the external specification. This type test is also called “**Black Box Test**”, because the tests are carried out without any internal information of the target software* such as control flow.

Internal Specification Test: The test cases of this type test are created based on the internal specification. This type test is also called “**White Box Test**”, because the tests are carried out focusing internal structure of the target software*.

*: In unit test, the target software is each unit. In program test, the target software is each program. And so on.

3.2. External specification Test/Black Box test

In External Specification Test, test cases are designed based on the external specification of a targeted unit, program or system.

The test cases should cover to test the followings;

- all functions which are specified in the external specification.
- all cases which have significant different results.

When the test cases to meet the above conditions are carried out, it may seem that the perfect test are done. However..

If something is missing in the external specification, some parts are left not to be tested.

3.3. Internal Specification Test/White Box test

In Internal Specification Test, test cases are designed based on the internal specification* of a targeted unit, program or system.

*: internal structure such as control flow, data flow, code...

The test cases should cover to test the followings;

- all processing structure which are coded in a programming language.

Typical test coverage measures based on programming languages are listed below;

C0 measure: number of executed statements / number of all statements

C0 measure at 100% means "all statements are executed".

C1 measure: number of branches passed / number of all branches

C1 measure at 100% means "all branches are passed/executed"

This tests prevent statements/branches from being left as non-tested parts.

This tests cannot detect functions which aren't implemented.

3.4. Combination of Black/White Box test

How to carry out efficient and sufficient test .

At first, the external specification test are carried out.

All external specification test cases should be executed, monitoring the C1/C0 coverage measures.

In this stage, if all test cases are successful, we could say all external specifications specified in the external specification are implemented.

At second, the internal specification test are carried out.

Analyze remaining internal structure/coding to be executed.

Add test cases to execute the remaining code should be executed within the external specification, monitoring the C1/C0 coverage measures.

In this stage, if all test cases are successful and C1/C0 coverage measures meet to 100%, we could say

"all functions specified in the external specification are successfully implemented non any redundant codes are there".

3.5. Test Coverage and Software Testing

3.5.1 Test Case Generation of "Examination Judgement Program"

In the following program, generate the test cases based on C0 and C1 test coverage.

Program Title: "Examination Judgement Program"

Subject: Two subjects as Mathematics, and Physics

Judgement specification:

Scores of both mathematics and physics are more than 69.

or,

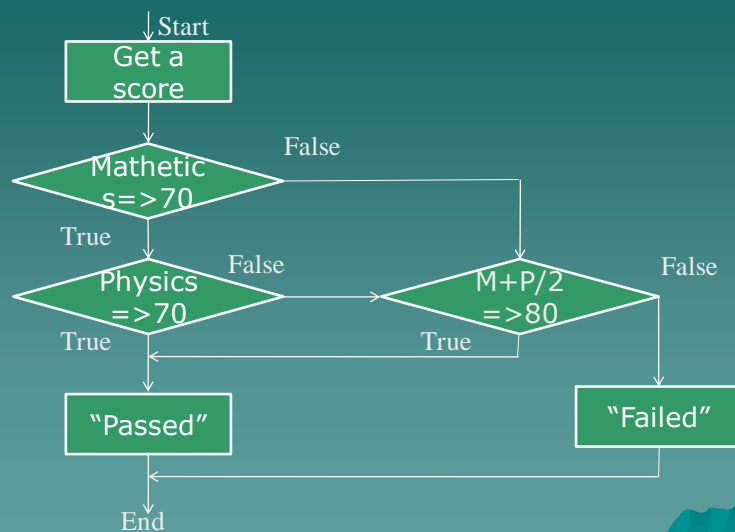
Average of mathematics and physics is more than 79.

→ "Passed"

otherwise

→ "Failed"

3.5.2 Flow Chart of "Examination Judgement Program"

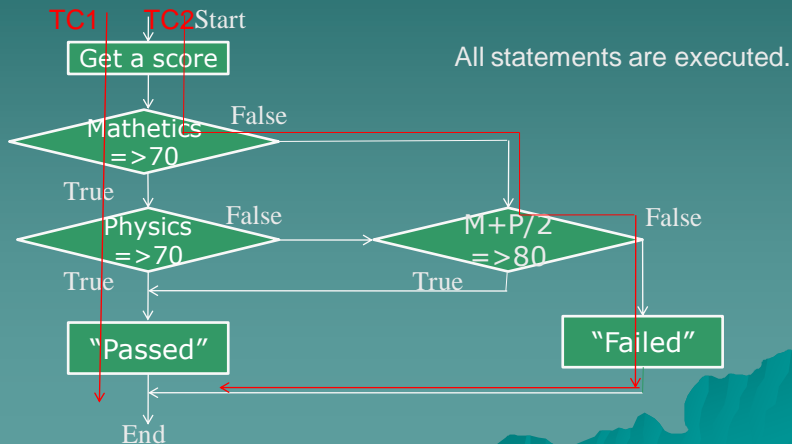


3.5.3. Test Cases based on C0 Coverage

For C0 coverage is 100%, need two test cases below.

TC1: Math score: 88, Physics score: 75 → "Passed"

TC2: Math score: 68, Physics score: 90 → "Failed"



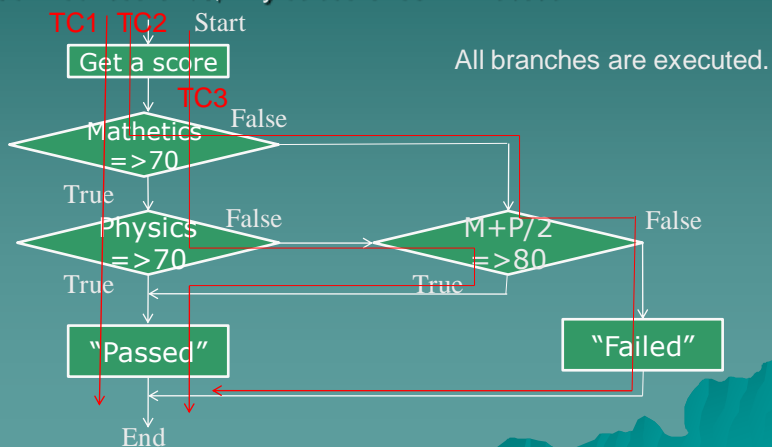
3.5.4. Test Cases based on C1 Coverage

For C1 coverage is 100%, need one additional test case TC3 below.

TC1: Math score: 88, Physics score: 75 → "Passed"

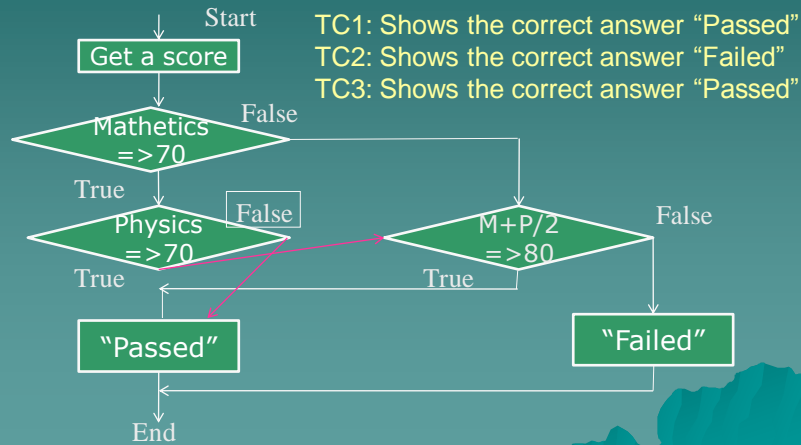
TC2: Math score: 68, Physics score: 90 → "Failed"

TC3: Math score: 98, Physics score: 68 → "Passed"



3.5.5. Analysis and Evaluation

Are these test cases enough to detect branch destination defects.
For example, the programmer make a simple mistake. Branch destinations of If “Physics>=70” statement are specified in revers.



3.5.5. Analysis and Evaluation

- ◆ Why the defects are not detected?
Analyzing using **decission table**

Decision Table

- ◆ Relations between the conditions for and the contents of the processing are expressed in the form of a table
- ◆ A decision table is a tabular form tool used when complex conditions are combined.

Decision Table - Example

- ◆ The conditions for creating reports from employee files

| | | | | |
|-----------------|---|---|---|---|
| Under age 30 | Y | Y | N | N |
| Male | Y | N | Y | N |
| Married | N | Y | Y | N |
| Output Report 1 | - | X | - | - |
| Output Report 2 | - | - | - | X |
| Output Report 3 | X | - | - | - |
| Output Report 4 | - | - | X | - |

3.5.5. Analysis and Evaluation (2)

This is external specification analysis.

Condition1: Mathetics=>70

Condition2: Physics=>70

Condition3: Average of Mathetics, and Mathematics =>80

| | TC1 | TCA | TC3 | TCB | TCC | TC2 | TCD | TCE |
|------------|------|-------|-------|-------|-------|-------|-------|-------|
| Condition1 | True | True | True | True | False | False | False | False |
| Condition2 | True | True | False | False | True | True | False | False |
| Condition3 | True | False | True | False | True | False | none | False |
| "Passed" | Yes | Yes | Yes | --- | Yes | --- | --- | -- |
| "Failed" | --- | --- | --- | Yes | --- | Yes | Yes | Yes |

TC1 has to detect these defects!

However, if the values of Condition2 change in reverse, TC1 results in the same. TC1 is not Condition2 sensitive. If TCA is chosen instead of TC1, it is condition2 sensitive!

3.5.5. Analysis and Evaluation (3)

This analysis is based on external specification. If we can assume that there aren't two defects at the same, TC1, TCD, TCE aren't needed.

Why do not TCD, TCE needed?

→ If there is a defect of Condition1 or Condition2, they cannot detect it. If there is a defect of Condition3, the defect can be detected by TCB or TC2.

→TC1, TCD, TCE aren't needed.

→In external specification viewpoint, we need only TCA, TC3, TCB, TCC, TC2.

Content

1. Testing Strategy
2. Testing Techniques
3. Statistical testing method
- ⇒ 4. Testing Method for Case Study
5. Other viewpoint for Test
6. Test Planning
7. Whole Testing Procedure in a Project

25

25

Use Case Based Test for System Test

To create the test cases based on UC "**Register for Course**" of the case study for external specification test.

In the UC "Register for Course", there are a main success scenario and 3 alternative flows named 4a, 5a, and 6a.

Alternative flow 4a: The student is not satisfied the prerequisite subject to the course to be registered. In this case, the system notifies the error message and return to the course registration panel.

Alternative flow 5a: Time of to be registered course is the same as the already registered course. In this case, the system notifies the defect message and return to the course registration panel.

Alternative flow 6a: 30 students already registered the course to be registered. Exceed the capacity of the course. In this case, the system notifies the error message and return to the course registration panel.

Use Case Based Test for System Test (2)

The UC scenario capable the followings interpretations as external specification;

1. Prerequisite subject check, time duplication check, and capacity check are dealt individually
2. Prerequisite subject check, time duplication check, and capacity check are dealt in this order.

Base on this interpretations, four test cases are designed as test cases executed all main/alternative flows;

TC.A: Successful case.

TC.B: Prerequisite error case.

TC.C: Time duplication error case. – No prerequisite errors are occurred!

TC.D: Capacity error case. – No prerequisite and no time duplication errors are occurred!

Use Case Based Test for System Test (3)

Will thses be enough test cases?

If the the interpretations are not correctly implemented?

→What test cases are needed to detect this types of errors?

→At all, are those errors such as check are done in deferent order?

Each checks has various conditions internally?

→Should this be covered by internal specification test?

Prerequisite subject check should be done first!

The check means if the student has the right to study the subject or not.

Therefore, if the system checks time duplication or capacity when the student has no right, the system become loaded with no-meaning process. And system **performance evaluation** is based on the correct sequence.

Also the student shloud be informed he/she is not satisfied the prerequisite. So, this check sequence is one of the **external specification**.

Use Case Based Test for System Test (4)

How to check the sequence? Tidy up using decision table.

There are 3 conditions below;

Condition1: The prerequisite subject are satisfied.

Condition2: The time is free without duplication.

Condition3: The capacity have leeway.

| | TC1 | TC2 | TC3 | TC4 | TC5 | TC6 | TC7 | TC8 |
|--------------|-----|--------|--------|--------|--------|--------|--------|--------|
| Condition1 | Yes | Yes | Yes | Yes | No | No | No | No |
| Condition2 | Yes | Yes | No | No | Yes | Yes | No | No |
| Condition3 | Yes | No | Yes | No | Yes | No | Yes | No |
| Registration | OK | --- | --- | --- | --- | --- | --- | --- |
| PreR. Error | --- | --- | --- | --- | Perror | Perror | Perror | Perror |
| Dup. Error | --- | --- | Derror | Derror | --- | --- | --- | --- |
| Capa. Error | --- | Cerror | --- | --- | --- | --- | --- | --- |

Use Case Based Test for System Test (5)

TC4, TC6, TC7, TC8 are the test cases which checks the sequence of the checks.

Next question is all test cases are needed?

If the sequence check isnot needed, the number of the test cases can be reduced four!!!

That can mean that TC1, TC2, T3, TC5 are more imortant!!

Content

1. Testing Strategy
2. Testing Techniques
3. Statistical testing method
4. Testing Method for Case Study
- ⇒ 5. Other viewpoints for Test
6. Test Planning
7. Whole Testing Procedure in a Project

31

31

5.1. Regression Test

If the software are made any changes or modifications, developer should carry out the re-test to confirm there are no side-effects.

The test is named "Regression Test".

For every bugs correction, the regression test which takes much effort and time is needed.

So, in "Regression Test" field, it is made much effort for automation testing.

5.2. Non-functional Test - Usability

As for the usability, in some stages/processes, it should be checked.

However, final usability test is achieved after the product developed.

The method observing the initial users to operate the new system is often adopted. In this case, sometime exclusive equipment and exclusive test/analyzed team carried out the test with customers.

5.3. Non-functional Test - Performance

As for the time efficiency/performance, in some stages/processes, it should be checked.

However, final time efficiency/performance test is also achieved after the product developed.

In real operation environment, highly-loaded test in which the system are burdened based on requirements are carried out.

Often the test is executed by supported tools or with customers.

Content

1. Testing Strategy
2. Testing Techniques
3. Statistical testing method
4. Testing Method for Case Study
5. Other viewpoint for Test
- ⇒ 6. Test Planning
7. Whole Testing Procedure in a Project

35

35

6.1. Master Plan for Testing

In upper stream process, detailed test should be planed!

For each test stage of unit test, program test, and system test, the test plan should described, in which listed below items can be included.

→Of course, they are depending on the software/system developed.

- Test targeted software
- Test schedule and method adapted such as bottom up or top down.
- Test Case Design method or policy including how many test case.
- Test environment to execute test cases.
- Test team/member to execute and manage.
- The plan and the policy for regression test, non-functional test.
- The plan and the policy for software qualification test.
- The management policy of all test phase/process

6.2. How many defects are detected in each test

“To detect more than 70% of the defects in code review and unit test” can be a good slogan. The test should be planned based on that concept.

-If most defects are detected in upper testing stream such as code review and unit test, it is not so difficult to make its quality stable because the quality of each unit is good in program test or later.

-If defects based on coding miss are existed in program test or later, it is not efficient, and it takes much time to test the program.

→Test plan should include the how many defects are detected in code review/unit test, in program test, and in system test.

6.3. Test Plan for each Software Component

To design the test plan for each software components considering the followings components condition;

- Function Type: user interface, Control logic, or DB operation....
- Function size and complexity
- Required quality

The test plan for each software components include the following items;

- Test Case design method: ex. Design based on State Transition Diagram→*
- The targeted number of test case: ex. 20 test cases/... steps→*
- The targeted number of detected defects in each test such as unit test, program test...
: ex. 15 in unit test, 5 in program test, 1 in system test→* use quality analysis later
- The test environments required.
- The test schedule...

Content

1. Testing Strategy
2. Testing Techniques
3. Statistical testing method
4. Testing Method for Case Study
5. Other viewpoint for Test
6. Test Planning
- ⇒ 7. Whole Testing Procedure in a Project

39

6.1. Grand Design for the Test of the Project-Example

| | Unit Test | Integrated Test | System Test |
|------------------------|------------------------------------|---------------------------------|-----------------------------------------|
| Who does? | Programer | Programmer/Tester | Test team/QA team |
| Place is | Each PC for development | Tset Environment | Pseudo Real Environment |
| Targets are | method, class | Program/ Component | System/ all programs |
| Test Sorts and /or | - External Spec.t. with CEGraph | - User Screen Interface test | -UC/UCS based test |
| Test Methods | -White box test with Coverage | -DB conect test | -Load performance t. -Usability test |
| Tool/ Qualification | -Test Coverage test tool | -Captuer and replay | -SW - Over load tool |
| Test Env. | -Test bed for UT | -DB Connection | - Usability test env. |

6.1. Grand Design for the Test of the Project-Example (2)

This is the example for the case study project.

There are various projects which have other pattern of test.

- Usually in software integration process, integrated test and system test are carried out. And then software qualification test is carried out just before the delivery. However, sometimes final system test carried out in customer site with the other system components.
- Testing Methods such as decision table, Case-Effect Graph and test coverage can be used in any tests. In the example table, typical usages are shown.
- Example schedule is shown in Gantt Chart example.

6.2. Grand Design for the Test – Approach

How to proceed the test in integration test?

In integration test, there are top-down approach and bottom up approach.

Top down approach: From upper modules such as control modules, the integration test is started. The advantage is that the test for the control modules are carried out, so early, then defects based on misunderstanding of specification can be detected early. However, test stub(s) which are dummy lower modules should be developed.

Bottom up approach: From lower modules such as data process/calculation module, the integration test is started. The advantage is that the lower modules are independent, so the tests can be carried out independently and on a parallel. However, test driver(s) which are dummy upper modules should be developed.

6.2. Grand Design for the Test – Approach (2)

For the integration test of software that has a certain level of size, each component is usually tested individually. And the test plan /procedure for each component should be constructed considering below;

- Functions/sub-components which are required high quality, or which have a significant impact on others, should be tested early.
- The test should be carried out in order growing efficiency of the test. For example, functions/sub-components which can be stubs or drivers for others should be tested early.

Here, top down approach using test stubs, and bottom up approach using test driver are used in combination.