4. Test Design Techniques

Hans Schaefer
hans.schaefer@ieee.org
http://www.softwaretesting.no/

Contents

How to find Test Conditions and Design Test Cases

Test requires concrete instructions.

Three work steps:

1. Test condition
2. Test case
3. Test procedure and plan for execution

Terminology

Test design specification: A document specifying the test conditions (coverage items) for a test item, the detailed test approach and identifying the associated high level test cases. [After IEEE 829]

Test condition: An item or event of a component or system that could be verified by one or more test cases, e.g. a function, transaction, quality attribute, or structural element. (Something to be tested).

Test case: A set of input values, execution preconditions, expected results and execution postconditions,

developed for a particular objective or test condition, such as to exercise a particular program path or to verify compliance with a specific requirement. [After IEEE 610]. A test case should be traceable to its requirements!

Example Test Design Specification

Testing the user interface of www.nsb.no

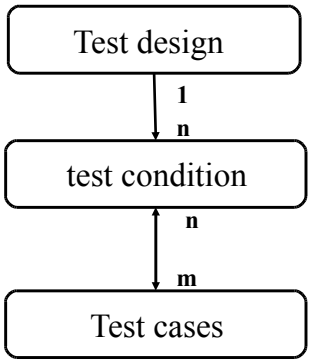
1 - is searching for trains part of the home page?
2 - single direction search?
3 - return trip search
4 - search between Norway and neighbor countries?
5 - ticket selling
6 - exception handling

Example Test Case

What: Searching connections between Norway and neighbor countries
Preconditions: Test client connected to www.nsb.no server, displaying home page.
Inputs:
- From stations: Bergen, Oslo, Trondheim, Narvik
- To stations: Stockholm C, Gøteborg C, Kiruna, Sundsvall
- Date and Time: Next Monday 08:00h

Expected results: Existing train connections (to be checked through www.bahn.de)
Postconditions: Test client still connected to www.nsb.no and no change in server database content.

Context



Test Cases

- What is tested here (test conditions)
- Preconditions
- Input
- Expected results
- Postconditions

Expected results
Discussion:

Have you ever checked that your bank correctly computes your account balance?

How to test this?

See next slide.

Expected Results - How Important?

Aren't they given by themselves?

Expected result = "The program behaves reasonably".

Problems if you do not specify:

- Under time pressure you overlook failures (everything is plausible).
- During test design you do not think through the specification.
- You forget to check results which are not easily visible (memory leaks, buffer corruption, background data, ...)

What is it?

- The results you should get, judging from the specification of the test object

Test Case Steps (not for exam)

1. Set-up: Acquire necessary resources. Instantiate the unit under test and put it into the desired state. Maybe require any other (surrounding, helper) units and test data.
2. Input: Send inputs to the unit under test.
3. Verify: Fetch the result of the stimulus. This may be output data or state changes. Collect these and return them to the test framework, which then reports the outcome to the tester.
4. Teardown: Release all the resources used, in order to avoid interdependencies between tests.

Test Procedure

How to run the test?

Test procedure: A document specifying a sequence of actions for the execution of a test. Also known as test script or manual test script. [After IEEE 829]

How Detailed Test Procedure?

It depends on tester skills and knowledge:

Worst case (test robot): Everything ”by spoon” - program, test script

Normally: Specify

- **order of test cases**
- **whatever needs special attention**

Best case: Only preparation and cleaning up is specified. Otherwise ”just run the tests”.

It also depends on requirements to documentation.

Test Procedure – IEEE 829-1998

.1 Test procedure specification identifier (Title)
.2 Purpose
.3 Special requirements
.4 Procedure steps: Include the steps in .4.1 through .4.10 as applicable.
- .4.1 Log (how to)
- .4.2 Set up
- .4.3 Start
- .4.4 Proceed
- .4.5 Measure
- .4.6 Shut down
- .4.7 Restart
- .4.8 Stop
- .4.9 Wrap up
- .4.10 Contingencies

What To Do During Test Design?

Analyze what to test.
-> test conditions
Translate test conditions to test cases with test data.

Traceability is important: Which test cases test what? Against specifications and requirements (->impact analysis, requirements coverage)
Choose good names for your test cases, put them into a data base, test management tool, ...

Risk considerations:
What is most important to test?
What is going to be changed most?

Techniques for Test Design

Black box - Specification based
White Box - Structure based: Everything in the code
Experience based - Tester's experience with bugs
Random test

We have to choose between those!

Reference: A Process for unit test design:
http://www.ipl.com/pdf/p0829.pdf

Black Box-Test Techniques

Black box = based on specification
Not based on code
May be the component specification

Finds especially forgotten or misunderstood items.

More about Black Box

Specification may be

- Models (formal, informal)
- Descriptions
- User manuals
- Our thinking about the system and its components

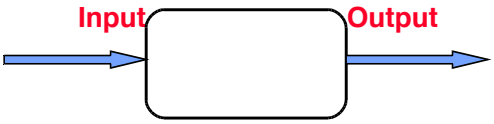
Try to cover the ”whole” specification.

Black box- techniques can be used BEFORE implementation!

White Box-Test Techniques

White box = based on the code
Everything in the code shall be executed
Test result taken from specification
Best on component level

Also: structure based, glass box, open box



Finds superfluous code. Finds ”holes” in the test. Important for security & safety requirements.

More about White Box

Start from
- Code
- Design and interfaces

Try to cover the ”whole” implementation.

Measurement of coverage possible by automated tools.

Many possible criteria.

Coverage can be checked, and then more test cases can be designed to cover “holes”.

Experience Based Test

Test knows typical problems.

Focus on:
- Typical faults
- Pessimism
- Unclear items
- Special values
- Special connections and contexts
- Risk factors
- Use of the program and its environment

Error guessing
Fault attack
Exploratory testing

Very dependent on knowledge, skill and experience!

Good addition to black and white box!

Random Test

Really black box
Random input, random-generated
Usage profile can be base for ”statistical test”

Finds problems not findable by systematic test design methods!
Example: “Fuzzing”. See http://www.codenomicon.com/free-ftp-suite/

Some Practical Questions

Plan for Execution / Test Procedure: How the Tests Are Executed

Test cases shall be executed. Normally, this requires a plan. Like a plan for film shooting.

Think about:

- Installation
- ”Smoke test”
- Main test
- New test after changes (regression test)
- Automatic or not
- Executed by whom, when, how, where
- Priority
- Dependencies between test cases

test execution schedule

How To Choose Techniques

Factors to consider:

Always consider several techniques depending on:

- Risk
- Expected types of faults
- Tester knowledge and experience
- Test level
- Tools
- Type of software, application or problem
- Customer requirements
- Requirements from authorities, contract, third parties

How To Choose Techniques

Black box

- Always. This is the use of test-driven design or the V-model.
- Finds misunderstandings early.
- Finds unclear specifications.

White box

- Especially for low level test.
- Tools available for nearly every programming language.
- A cheap insurance.

How To Choose Techniques

Experience based test

- Can be very effective
- Requires experience :-)
- Does not replace black box-methods

Random test

- Not used often
- Good against problem or project blindness
- For very high reliability applications

Heuristics for test conditions (not for exam)

Touring Heuristic
Mike Kelly on Tue, 20/09/2005 - http://www.testingreflections.com/node/view/2823
Mnemonic:

FCC CUTS VIDS

The mnemonic stands for the following:

Feature tour
Complexity tour
Claims tour

Configuration tour
User tour
Testability tour
Scenario tour

Variability tour
Interoperability tour
Data tour
Structure tour

*** Feature tour: Move through the application and get familiar with all the controls and features you come across.**
*** Complexity tour: Find the five most complex things about the application.**
*** Claims tour: Find all the information in the product that tells you what the product does.**
*** Configuration tour: Attempt to find all the ways you can change settings in the product in a way that the application retains those settings.**
*** User tour: Imagine five users for the product and the information they would want from the product or the major features they would be interested in.**
*** Testability tour: Find all the features you can use as testability features and/or identify tools you have available that you can use to help in your testing.**
*** Scenario tour: Imagine five realistic scenarios for how the users identified in the user tour would use this product.**
*** Variability tour: Look for things you can change in the application - and then you try to change them.**
*** Interoperability tour: What does this application interact with?**
*** Data tour: Identify the major data elements of the application.**
*** Structure tour: Find everything you can about what comprises the physical product (code, interfaces, hardware, files, etc…).**