Introduction to Systematic Testing II

Outline

- Last time we began with basic definitions, validation & verification, the role of specifications, and the levels of testing
- Today we continue with:
  - Testing in the life cycle and the different kinds of testing
    - Introduction to testing methods: black box & white box
  - Test design and strategy
  - Test plans and procedures
  - Test results

Testing in the Software Life Cycle

Kinds of Tests

- Testing has a role at every stage of the software life cycle
- As we have seen, tests play a role in:
  - the development of code (unit testing),
  - the integration of the units into subsystems (integration testing) and
  - the acceptance of the first version of the software system (system testing)

Testing in the Software Life Cycle

Kinds of Tests

- We will divide these tests into:

Black Box Testing Methods

- Black box methods – cannot see the software code (it may not exist yet!) – can only base their tests on the requirements or specifications

White Box Testing Methods

- White box (aka glass box) methods – can see the software's code – can base their tests on the software's actual architecture or code

Testing in the Software Life Cycle

Regression Tests

- In addition, as the system is maintained, other kinds of tests based on past behaviour come into play
- Once a system is stable and in production, we build and maintain a set of regression tests to ensure that when a change is made the existing behaviour has not been broken
- These often consist of a set of actual observed production inputs and their archived outputs from past versions of the system

Testing in the Software Life Cycle

Failure Tests

- As failures are discovered and fixed, we also maintain a set of failure tests to ensure that we have really fixed the observed failures, and to make sure that we don't cause them again
- These consist of a set of actual observed inputs that caused the failures and their archived outputs after the system is fixed

Test Design

Design of Tests

- The design of tests for a system is a difficult and tricky engineering problem as important as design of the software itself
- The design of effective tests requires a set of stages from an initial high level test strategy down to detailed test procedures
- Typical test design stages are:
  - test strategy, test planning, test case design, test procedure

Test Design

(1) Test Strategy

- A test strategy is a statement of the overall approach to testing for a software development organization
- Specifies the levels of testing to be done as well as the methods, techniques and tools to be used
- Part of the project's overall quality plan, to be followed and reported by all members of the project

Test Strategy Examples

Big Bang Testing Strategies

- Test the entire software once it is complete.

Incremental Testing Strategies

- Test the software in phases (unit testing, integration testing, system testing)
- This testing strategy is what we use in Agile Development
- Incremental testing can occur bottom-up (using drivers) or top-down (using stubs)
- In general bottom-up is easier to perform but means the whole program behavior is observed at a later stage of development

Test Strategy Examples

Big Bang vs. Incremental

- Big bang testing only works with a very small and simple program
- In general, incremental testing has several advantages:
  - Error identification ✓
    - Easier to identify more errors
  - Error correction ✓
    - Simpler and requires less resources

Test Design

(2) Test Plans

- A test plan for a development project specifies in detail how the test strategy will be carried out for the project
- In particular, it specifies:
  - the items to be tested
  - the level they will be tested at
  - the order they will be tested in
  - the test environment
- May be project wide, or may be structured into separate plans for unit, integration, system and acceptance testing

Test Design

(3) Test Case Design

- Once we have a plan, we need to specify a set of test cases for each item to be tested at each level
- Each test case specifies how the implementation of a particular functional requirement or design unit is to be tested and how we will know if the test is successful
- It is important to include test cases to test both that the software does what it should (positive testing) and that it doesn't do what it shouldn't (negative testing)
- Test cases are specified separately at each level: unit, integration, system and acceptance – and their documentation forms a test specification for the level

Test Design

(4) Test Procedures

- The final stage of test design is the test procedure, which specifies the process for conducting test cases
- For each item or set of items to be tested at each level of testing, the test procedure specifies the process to be followed in running and evaluating the test cases for the item
- Often this includes the use of test harnesses (programs written solely to exercise the software or parts of it on the test cases), test scripts (automated procedures for running sets of test cases), or commercial testing tools

Test Reports

Documenting Test Results

- Output of test execution should be saved in a test results file, and summarized in a readable report
- Test reports should be designed to be concise, easy to read and to clearly point out failures or unexpectedly changed results
- Test result files should be saved in a standardized form, for easy comparison with future test executions

Systematic Testing Methods

Systematic Methods Recap

- Recall that to be a systematic test method, we must have
  - a system (rule) for creating tests
  - a measure of completeness
- Need an easy, systematic way to create test cases
  (to know for sure what to test)
- Need an easy, systematic way to run tests
  (to know how to test)
- Need an easy, systematic way to decide when we're done
  (to know when we have enough tests)

Summary

Introduction to Systematic Testing II

- Testing is not just a one-time task – it is a continuous process that lasts throughout the software life cycle
- Effective testing requires careful engineering, similar and parallel to the process for design and implementation of the software itself
- An overall test strategy drives test plans, test case design, and test procedures for a project
- Two classes of systematic test methods, black box and white box

References

- Galin ch. 9 (9.2)

Next Time

- Black Box Testing